# Institutions under composition

Joshua Zejun Tan

Magdalen College
University of Oxford

*A thesis submitted for the degree of*
*Doctor of Philosophy*

Trinity 2022

## Abstract

The main subject of this thesis is how compositionality, which appears everywhere in computer science and mathematics, can be applied to the design of social institutions such as contracts, courts, and digital platforms. Our analysis draws on category theory, especially the theory of open games. But wherever possible, we have not only specified theories but attempted to test those theories in practice. Wherever possible, we have gone beyond toy examples to produce live examples and concrete tools that can make a difference to the people that use them. The essential thing is not any particular representation of institution: the particular representation will vary depending on the application, as will the particular model of composition. The essential things are the patterns in which people interact.

# Institutions under composition


Joshua Zejun Tan

Magdalen College

University of Oxford

A thesis submitted for the degree of

*Doctor of Philosophy*

Trinity 2022

To a schmoo.

For by art is created that great LEVIATHAN called a COMMONWEALTH, or STATE, in Latin CIVITAS, which is but an artificial man...

— Thomas Hobbes, *Leviathan*

In mathematics, there are not only theorems. There are, what we call, "philosophies" or "yogas," which remain vague. Sometimes we can guess the flavor of what should be true but cannot make a precise statement. When I want to understand a problem, I first need to have a panorama of what is around it. A philosophy creates a panorama where you can put things in place and understand that if you do something here, you can make progress somewhere else. That is how things begin to fit together.

— Pierre Deligne

# Acknowledgements

Thank you to my supervisors, Samson and Bob.

Thank you to my many mentors and collaborators, including Prima, Nathan, Larry, Seth, Amy, Philipp, Jules, Megan, Zargham, Luke, Ellie, Divya, Isaac, Brendan, Nina, David, Scaz, Carol, Sylvain, Misha, and many, many others. And to the institutions and communities that made this work possible—the Quantum Group at Oxford, Magdalen College, Harvard Law School, the Fleet fellowship at Princeton, the Stanford Digital Civil Society Lab, Heinz College at Carnegie Mellon, and especially Metagov—thank you too.

And for all your love and support—thank you Mom, Dad, Stephen, Yeiyei, Nainai, and, of course, Janina.

# Abstract

The main subject of this thesis is how compositionality, which appears everywhere in computer science and mathematics, can be applied to the design of social institutions such as contracts, courts, and digital platforms. Our analysis draws on category theory, especially the theory of open games. But wherever possible, we have not only specified theories but attempted to test those theories in practice. Wherever possible, we have gone beyond toy examples to produce live examples and concrete tools that can make a difference to the people that use them. The essential thing is not any particular representation of institution: the particular representation will vary depending on the application, as will the particular model of composition. The essential things are the patterns in which people interact.

# Contents

# List of Figures

# 1 Introduction

## Contents



The year is 1920. The Great War is finally over. Automobiles are everywhere. There's electric light, Fats Waller on the radio, the Jazz Age. And science too is on the march. Not just physics or medicine, but the study of society. In 1920, the social sciences as we know them today are just being invented. Econometrics is young, organization studies is in its infancy, America has just graduated its first MBAs, and at the vanguard of this era is a man named Roscoe Pound, who has just been made the dean of Harvard Law School.

Now Pound was a man of unwavering curiosity, a man with a PhD in botany who had published nine academic papers by the age of 19, and he believed that law could be made into a kind of engineering and that lawyers could be trained as social engineers [1]. "By social engineer I mean, on the analogy of the industrial engineer, one whose calling it is to make a social process or activity achieve its purpose with a minimum of friction and waste" [2]. To be clear, Pound was not saying that law could be turned into a precise mathematics. There was no formula that, after

putting in the facts of the case, would simply produce the "right" judgment or the right law. But this did not preclude a systematic, engineering approach to law.

For Pound, the law has a proper end. That end is to maximize the welfare of society. And in the service of that end, any particular legal or contractual intervention ought to be methodically optimized. Moreover, science has a role to play in the law, especially the social sciences. If the end of the law is to satisfy the interests of society while minimizing different forms of social "friction", then law ought to respond to society as it actually was. Lawyers and legislators thus had a responsibility to learn the science of society, and to design laws and contracts which respond to that science.

So in 1920 we already have a surprisingly radical question: can we "engineer" a law, given a scientific description of society?[1] Later on we'll ask a similar question: can we engineer institutions, given cyberspace?[2] But for now the problem is: can we engineer laws, given society? For simplicity, assume that we get one chance to set the law, without caring about what the law looked like in the past or how the law might evolve in the future.

In that case, the answer is no. A lawyer or a judge today cannot just look at a legal problem and solve it like an engineer. This is not a sharp or essential difference between the two fields but a pragmatic one. Lawyers can already tinker with laws and contracts and arguments—make small changes in penalties, funding, enforcement, implementations, purview, rationales, loopholes, exceptions, and so forth—but what distinguishes a practice as "engineering" from well-informed tinkering is the presence of a scientific system to reason about these changes and their likely effects. As for the development of such a scientific system in the law, there is progress in parts: certain aspects of the law like monetary policy or spectrum auctions have evolved into a kind of economic engineering [3]; there is an emerging discipline around the engineering of blockchains and smart contracts [4] (though these rarely involve "law"

[1] I am taking some liberties with Pound's language in order to make the question more relevant for engineers. Pound is addressing a community of lawyers and jurists, and his recommendations are mostly philosophical or policy-oriented.

[2]

per se); contract theory uses game theory to model particular aspects of contracts [5, 6]; and many different aspects of social science—from operations research to sociocybernetics to management science to public economics—have tried to apply science and mathematics to make administrative policy more predictable, more effective, and more accountable [7–9]. But in the vast majority of cases, writing a law or a contract is still not like designing a spring, a pump, or an engine.

## 1.1 Institutions under composition

Laws are a kind of *institution*, and we want to engineer institutions. For now, think of an institution as just a system with a number of agents, with whatever intuitions you wish to attach to 'system' or 'agents'. In what follows, we will almost always view institutions prescriptively, as distinct subsystems which regulate or intervene on the behavior of agents who might otherwise be doing something else. On this view, a control module is an institution, as is a law. A game of hide-and-seek is an institution, one that can be played by a schoolyard of kids or by a set of potato-headed virtual avatars [10]. Markets are a kind of institution, one which is usually modeled in terms of demand curves, supply curves, and price mechanisms. A voting regime is another kind of institution, one which is often modeled by preference aggregation functions which in turn depend on other models.

Broadly speaking, the main subject of this thesis is how compositionality, which appears everywhere in computer science and mathematics, can be applied to the design of social institutions such as contracts, courts, and digital platforms. In the first part of this thesis (Chapters 2-3), we will model institutions as games, especially open games in the sense of compositional game theory [11]. While more scalable and less small-bore than many existing approaches, these chapters still focus on the modeling and implementation of *single* institutions: auctions for carbon credits, monitoring arrangements for Nepalese farmers, investment contracts for Silicon Valley startups. In the second part of this thesis (Chapters 4-6), we develop a web-scale approach to institutional engineering, emphasizing the development of tools, infrastructure, and standards that (1) help people build their own digital

institutions and (2) define ways in which those institutions can interact. The pivot to an online setting is especially important in light of limitations to compositional game theory raised in the first chapters. Both parts of the thesis owe much to the work of Elinor Ostrom and her collaborators in institutional analysis and design (IAD) [12], which we review in Chapter 4. Finally, in the conclusion we propose a research program aimed at accelerating institutional innovation that goes beyond the limitations imposed by compositionality itself.

This thesis invokes and references many theories of "institution": as "enduring regularities of human action in situations" and/or as formal grammars within an ethnographic study [13] in Chapter 6, as families of game forms [14] in Chapter 2, as settings within the study of human-computer interaction [15] in Chapter 5, as action situations [16] in Chapter 2, Chapter 3, and Chapter 4, as modalities of regulation [17] in Chapter 4 and Chapter 6, as dynamical patterns of behavior [7, 8] in Chapter 2, and as structures of governance broadly understood in Chapter 4 and Chapter 6. It is often impossible to avoid mixing these theories in practice, a fact which we regard as both inevitable and beneficial. Wherever possible, we have not only specified theories but attempted to test those theories in practice. Wherever possible, we have gone beyond toy examples to produce live examples and concrete tools that can make a difference to the people that use them. The essential thing is not any particular representation of institution: the particular representation will vary depending on the application, as will the particular model of composition. The essential things are the patterns in which people interact.

## 1.2 Thesis outline

The remainder of this report is organised as follows:

> Chapter 2 — we illustrate the benefits of compositional game theory to institutional design via three case studies, with code. Based on joint work with Seth Frey, Jules Hedges, and Philipp Zahn, whose content has been published in PLoS One [18].

Chapter 3 — we present Clause2Game, a tool for modeling clauses and boilerplate in contracts. Based on joint work with Philipp Zahn and Megan Ma, whose content has previously been presented at Computational Legal Studies 2022.

Chapter 4 — we present a conceptual proposal for a governance layer for online communities. Based on joint work with Nathan Schneider, Seth Frey, Primavera De Filippi, and Amy Zhang, whose content was presented at CSCW 2021 [19].

Chapter 5 — we analyze digital agreement systems, and present the Agreement Engine, a tool for building modular agreement systems. Based on joint work with Luke Miller, whose content has been published in the MIT Computational Law Report [20].

Chapter 6 — we present decentralized autonomous organizations (DAOs) on the blockchain as a model of a governance layer for online communities, and discuss ongoing work to build interoperable standards and infrastructure for the DAO ecosystem. Based on joint work with Isaac Patka, Michael Zargham, Ido Gershtein, Eyal Eithcowich, and Sam Furter, whose content has been published on Ethereum.org [21], as well as forthcoming work with Tara Merk, Sarah Hubbard, and Eliza Oak.

Chapter 7 — we conclude by reflecting on institutions within artificial intelligence, which suggest two directions for future work.

## 1.3 Statement of collaboration

Chapter 2 is based on a joint collaboration with Seth Frey, Jules Hedges, and Philipp Zahn, while Chapter 3 is based on follow-up work with Philipp Zahn and Megan Ma. In the first collaboration all parties contributed equally to the paper, working across examples, narrative, and editing. In the second collaboration, I led the conceptualization, writing, and supported some of the coding of the

examples. Chapter 5 is a joint collaboration with Luke Miller in which I led the conceptualization and writing while Luke led the development of the code examples. Chapter 4 is a joint collaboration with Nathan Schneider, Seth Frey, Primavera De Filippi, and Amy Zhang in which I was the primary instigator with Nathan Schneider; I developed most of the technical elements, diagrams, and contributed substantially to the conceptualization, background, and examples. The contents of Chapter 6 are based on a joint collaboration with Isaac Patka, Michael Zargham, Ido Gershtein, Eyal Eithcowich, and Sam Furter as well as a separate collaboration with Tara Merk, Sarah Hubbard, and Eliza Oak; in both of those collaborations I was the lead author working across all aspects of the conceptualization and writing.

# 2

# From games to institutions

How can we build larger models of institutions out of smaller ones? The primary contribution of this chapter is practical and methodological: we demonstrate how compositional game theory, grounded in the mathematics underlying programming languages and introduced here as a general computational framework, increases the parsimony of game representations with abstraction and modularity, accelerates search and design, and helps theorists across disciplines express realistic institutional complexity in well-defined ways. Relative to existing approaches in game theory, compositional game theory is especially promising for solving game systems with long-range dependencies, for comparing large numbers of structurally related games, and for nesting games into the larger logical or strategic flows typical of realistic policy or institutional systems.

## Contents



Game theory, since its development by mathematician von Neumann and economist Morgenstern [22], has proliferated through the biological and social sciences as a powerful formalism for modeling strategic and cooperative interactions.

Economics in particular has applied it to core disciplinary questions, with a keen interest in analytical modeling and the formal properties of game solutions. However, this wildly successful research agenda has obscured other promising uses of game theory. For instance, game theory has also long been recognized as a potential tool for the faithful description and detailed design of realistic social institutions [23]. Calls for this high-fidelity or "descriptive" game theory have been heard from disciplines as diverse as international development [24], law [25], animal behavior [26], institutional economics [27], and sustainability [28]. For example, political scientist Elinor Ostrom introduced the "action situation" framework as an empirically grounded generalization of game theory for structuring ethnographic description [12], and she imagined formal representations of institutions in terms of systems of linked action situations. The economist Leonid Hurwicz pursued the same conception of institutions as linked systems of games [14]. In these approaches, the central questions about an institution may not involve its solutions but the uniqueness of its decision structure or its structural complexity relative to comparable institutions.

These new uses require a scalability, heterogeneity, and overall complexity that existing game design formalisms struggle to capture. Within familiar normal- and extensive-form computational representations (typified by software libraries such as Gambit [29]), each additional player, choice, and stage added to a game contributes to an exponential growth of game outcomes, and a proliferation of equilibria. These threats to game expressiveness highlight the need for a theory of complexes of games that permits modularity, abstraction, and other core principles of engineering, particularly software engineering.

With the introduction of structured programming, and the formal apparatus of modern computer science generally, Edsger Dijkstra and other early researchers abstracted out of machine code to focus on higher-level questions of software architecture [30, 31]. For the same reasons that software engineers have adopted modern computer languages, we offer compositional game theory for designing complex institutions Fig. 2.0.1.

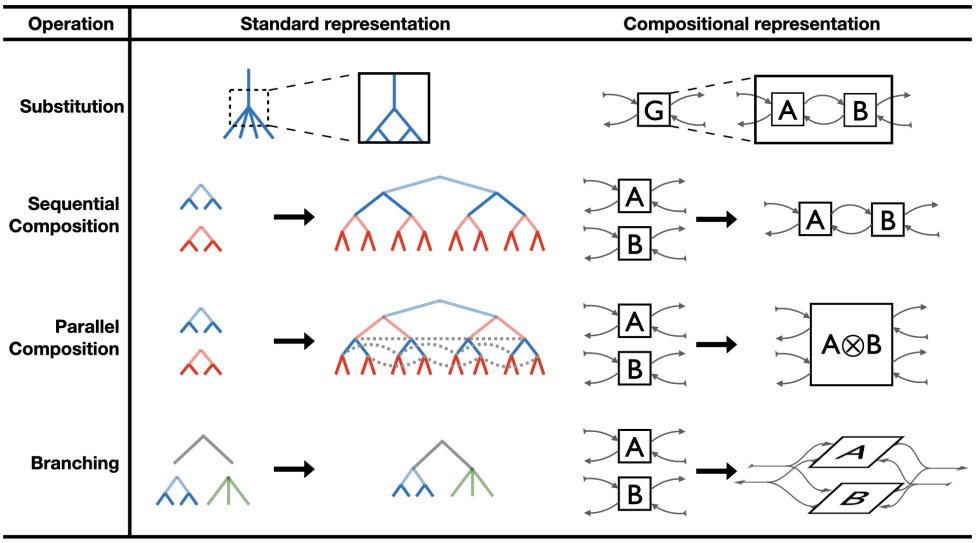



**Figure 2.0.1: Transforms illustrating the range of compositional game theory.** We show four illustrative types of abstract game transforms that, when combined, can produce complex institutional forms, toward a high-level, compositional language for linking games into larger systems. Each row shows an extensive form representation of the pattern, followed by a compositional string representation of that pattern. *Substitution* permits modularity and abstraction, the basis of a high-level hierarchical design approach for complex games. *Sequential composition* arranges games in series in a way that abstracts over specific game outcomes, which otherwise grow exponentially in large or repeated games. *Parallel composition* arranges games for simultaneous play in a way that abstracts out of the specifics of complex information sets (dashed lines). *Branching* allows an upstream decision to influence what games are played downstream. It can be seen as providing an XOR choice in contrast to the AND of the other two types of composition. In the cases below, #1 and #2 use substitution, all three use sequential composition, #1 and #2 use parallel composition, and #3 uses branching. With these and other transforms, compositional game theory provides a concise, unified language focused the high-level, architectural dimension of game-theoretic institution design.

The primary aim of this work is to organize prior work on compositional game theory for an audience beyond applied mathematics and theoretical computer science, with a particular focus on interdisciplinary and computational social scientists. Among social scientists, calls for complex games have come from every discipline, but have been most clear and consistent from environmental science and organizational and institutional analysis. Within applied mathematics and mathematical game theory, this work motivates continued formal development by communicating the diversity and importance of its applications.

Compositional game theory articulates traditional game theory in terms of

category theory, a branch of mathematics that has been used fruitfully to map software engineering concepts into domains such as quantum computing [32], chemistry [33], and natural language processing [34]. We show, across three cases, how compositional game theory can expand the scope of game theory while supplementing existing ethnographic and other empirical methods. Under the compositional framework, designers can nest games within each other, give players choices between games, and create complex logical flows across games or long-range dependencies within them. Compositional game representations abstract nonessential details of specific games to allow systems of games to be compared, allowing modelers to capture connections between game architectures, and how a game's solutions change with its embedding in different social or policy contexts. In this way, the compositional approach opens several subjects to more practical analysis: large chains of many games, comparisons of structurally similar games, complex logical/strategic flows through games (games of games), and the efficient interactive design of all of the above. It also supports a formal visual string diagram language, and permits designers to quickly prototype new architectures and map proven ones into new contexts. Game complexes are still compatible with existing solution concepts and proof methods, but the theory operates at a more abstract level that focuses modelers on the high-level work of composing and extending them.

To be explicit, this work does not contribute to game theory by offering new equilibria or faster solutions to existing equilibria, and its contribution to the examples we explore below is not to solve them. Construed broadly, classical game theory has no formal limitation that compositional game theory overcomes. And the compositional approach does not solve the problem that large complexes of games may have dozens or hundreds of solutions under familiar solution concepts. But merely introducing compositionality brings attention to the need for solution concepts that are selective enough to aid the analysis of very large games. Compositional game theory gives modelers and designers a framework for exploring and iterating over arbitrarily large games, but it is more than a representational advance. By extending the range of social systems that can practically be expressed game theoretically, through

improved designability, extensibility, comparability, and visualization, it is also a source of new questions about the game theoretic study of institutions.

## 2.1 Compositional game theory

Compositional game theory is a formal framework for composing economic games into larger systems of games [11, 35, 36]. It is grounded in category theory, especially the categorical approach to open systems [37, 38]. In technical terms, this approach models systems as morphisms $f : X \to Y$ in a symmetric monoidal category where the objects $X$ and $Y$ describe the boundaries of the open system. This is notable because of the connection it reveals between the structures of game theory and software architecture. Classic models of computation such as lambda calculus have been productively modeled using category theory [39]. As well as abstraction and modularity, open games admit a formal graphical representation [40, 41] that is closely related to other formal diagram systems, such as Feynman diagrams [42].

Framed within category theory, a game is a kind of process, following the arrow of time from past to future Fig. 2.1.1. And the basic unit of categorical game theory, the open game, generalizes a game so that it can communicate with an external environment through its inputs and outputs, which define its type. Open games connect along their type boundaries to compose into larger open games in such a way that each component becomes a part of the environment of the others. This is directly analogous to how modern software is built by connecting standard components—such as functions, classes, and modules—through well-defined interfaces. In fact, the analogy is direct enough that string diagrams of open games can be directly compiled to software and are subject to formal guarantees, such as the guarantee that any composition of open games will be another well-typed open game.

To give a sense of the expressiveness of this approach, we describe the composition operations and simple design patterns underlying the cases below. In the most familiar operation, we introduce *substitution*, in which an institutional process includes a black box that can be filled in with any game that produces inputs and outputs of the right types (Figure 2.0.1, Row 3). A system of games that satisfies

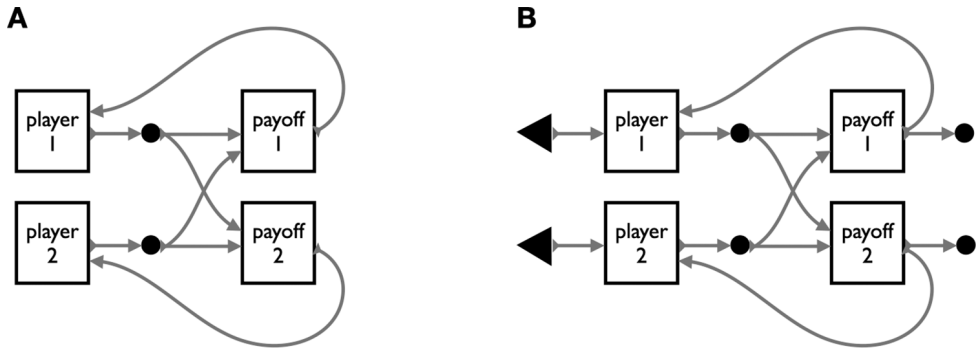



**Figure 2.1.1: Closed and open versions of the string diagram of an n-choice, two-player game.** Two players emit decisions based on prospective information about (only) their own payoffs. Their decision feeds into the calculation of their own payoff and that of the other player. Time proceeds left to right, with players making decisions that emit payoffs. Arrows feeding backwards in time to players represent player preferences over future events and signify the presence of strategic reasoning. In these diagrams, specific choice sets and payoffs are abstracted away, improving parsimony as games scale. Formal string diagrams of this style map directly to game architectures in the sense that the computational representation of a game could be compiled to or from its diagram. Boxes are general, and can be used to represent players, payoffs, decision nodes, entire games, or any potential target of substitution. **A**. This "closed" version of a game is consistent with conventional game theory. Players are fixed, and results of the game feed back to those players. **B**. The closed game can be opened with the addition of inputs and outputs, represented by incoming or outgoing arrows. In the open version, players are replaced by inputs of player type, enabling flexibility as to how agents are selected to fill the player role, and reuse of the game in different contexts. This version of the game also has open outputs. In addition to feeding payoff information back to the players, it emits them as outputs that could, for example, be used to parameterize a downstream game. We show this in Fig. 2.3.1, an irrigation social dilemma, which models the steady depletion of a water level variable by making the output of one decision unit the input of the next.

the same constraints can also be substituted, allowing modularity and abstraction. Second, in *serial composition*, two games are appended. This operation is intuitive in part because it has a clear analogue in extensive game forms (Figure 2.0.1, Row 1). But the weakness of the extensive representation—an exponential growth in outcomes with number of games—highlights the strength of our approach to serial composition, which abstracts out of specific payoffs and outcomes at the level of the user. Another operation we distinguish is *parallel composition*, in which several games are played simultaneously, perhaps with a complex information architecture (in terms of which player know what when; Figure 2.0.1, Row 2). Existing representations can capture the complex information sets that fall out of

the parallel composition operation but, again, these quickly become unwieldy with increases in the number of games being composed. A fourth design pattern we introduce is *branching*, which permits a game outcome to output not just numbers (payoffs), but pointers to games and players. With branching, the outcome of a game representing the policy design process can be a game representation of the designed policies. This list of four operations and design patterns is by no means exhaustive, but as a subset of possible patterns, they enable a broad range of game operations, as we show in three examples below.

## 2.2 Case 1: $CO_2$ certificate markets

*In this example, we give a common example from market design that illustrates the typical "zoom-in, zoom-out" modeling approach of an open game. This particular example uses substitution, sequential composition, and parallel composition.*

Markets for emissions are a prominent tool in the economic fight against climate change. As in the analysis of climate negotiations [43], game theory has played a crucial role in research on these sometimes complex institutions [44].

Institutional arrangements like emission markets are interesting because their many moving parts can interact in unexpected ways. Consider a simplified $CO_2$ certificate market, which involves an allocation stage for initially distributing certificates, a production stage in which players generate $CO_2$, a resale stage for trading partially and over-fulfilled certificates, and a second production stage Fig. 2.2.1. A researcher might ask several questions. How should the initial permits be allocated? How should the resale market be structured? What are the distributional effects on producers? How are consumers affected?

These questions hinge on the subtle, long-range interactions between stages of this market. For instance, details about the stage one allocation of permits can have an indirect effect upon the stage three resale market [45–47]. In this system, the allocation and resale stages are strategic (as indicated by the presence of backwards-facing arrows, signifying a decision that depends on prospects about

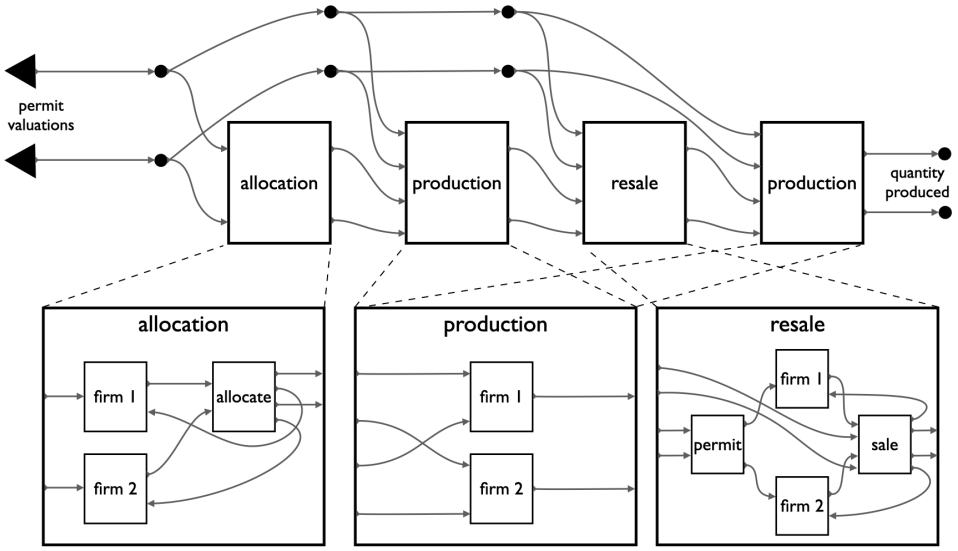



**Figure 2.2.1: A four stage $CO_2$ market game.**This multi-stage game proceeds through an initial allocation stage, a production stage, a resale stage, and a second production stage. The first models the primary allocation of $CO_2$ certificates to producers. Producers who received permits then decide how to use them in production. Afterwards, they either have unused permits left or are seeking further permits, and so participate in a resale market that is then followed by a final production phase. Producers operate under incomplete information: they do not know how highly others value their permits. With each stage represented as modules, stages like the production stage can be reused. The explicitly typed incoming (large left-pointing triangles) and outgoing arrows (terminating in circular nodes that represent the game's composability) make this complex of open games itself a game that could be opened and embedded within a larger game.

future interdependent outcomes). By contrast the production stages are non-strategic, in the sense that they can be made entirely on the basis of information that doesn't depend on the decisions of others. But once embedded between strategic decisions, production decisions begin to figure into a firm's strategic reasoning. Existing models of these markets often focus on one or two stages in isolation, resulting in a collage of models which succeed in analyzing different aspects but fail to provide a global, integrated view of the full process, or how it will play out differently in different contexts.

Compositional game theory offers an alternative modeling approach, one that brings to game theory capabilities for modularity, reuse, abstraction, and other principles of programming. Individual components are modeled with an interface relative to an environment. As with traditional models modelers can zoom in on

specific components and analyze them in isolation. They can also substitute parts in the process of modeling. In the compositional approach, the required interfaces constrain the modeling of each stage to ensure that it can communicate with the rest of the model. Modelers thereby gain a theoretical framework for iterating systematically through variants of a large connected system, while being able to freely switch between traditional equilibrium analysis and other methodologies like simulation.

## 2.3 Case 2: Nepali irrigation monitoring schemes

*In this example, we consider a classic example from the commons governance literature and use it to illustrate how compositional game theory can be scaled cleanly to a family of related institutions through an iterative, exploratory modeling flow. This particular examples uses substitution, sequential composition, and parallel composition."*

From fisheries to pastures, forests, and irrigation systems, communities around the world depend upon the successful community management of common-pool resources. But because communities differ greatly from each other, the principles of success can be elusive. This is especially clear in studies such as those by Ostrom and colleagues [48, 49], who compared the collective action institutions of hundreds of small-scale irrigation systems in Nepal.

These works examined institutional diversity, the range of successful approaches to a given collective action problem: when there is asymmetrical access to limited water by upstream farmers, "head-enders" can leave "tail-enders" with insufficient resources. Farmer communities have successfully evolved many different institutions for solving this dilemma, but they are difficult to compare with existing tools. Communities have been observed to rotate water access by season, crop, farmer, or day of the week. They may or may not rely on monitors to enforce local rules. Those that do might pay their monitor fixed fees, fractions of crop yields, or they might allow their monitor to administer and retain penalties. Represented as games,

farmer players can use water profligately or equitably, monitors can exert costly work or shirk, and each regime interacts with these choices differently.

With game modeling tools focused on fine-grained institutional features, it can be difficult to see the features that such diverse institutions have in common against the noise of their differences: differing numbers of players, numbers and types of choices, specific payoffs, and other factors. Kimmich, for example, models an irrigation governance system as a network of six adjacent games, to show how incremental changes in one part of the network ripple through to affect the equilibria elsewhere [23].

Using the compositional framework, we developed a simple grammar for capturing institutions in the Nepali irrigation case, to rapidly build and test different observed variants. Under a framework focused on architecture, designers and modelers can iterate efficiently while managing the cascading effects of structural changes. The result represents the range of regimes against the background of their structural similarities. One thing that becomes apparent is that the process of structuring incentives toward greater fairness requires increasing the complexity of the institution and decisions within it. As indicated by the monitor's backwards-facing arrows in Fig. 2.3.1, the variations with greater capacity for fairness are also those with a greater number of interacting strategic decisions.

We developed these variants through an iterative, exploratory process. One conclusion of this comparative modeling exercise was that we cannot engineer a unique Nash equilibrium that is equitable, with all farmers extracting the same amount, without carefully combining several types of incentives. The flexibility and extensibility of the compositional framework permitted us to come to this conclusion through a rapid and interactive exploratory design process. We started with the **B** game of Fig. 2.3.1 and ideas for possible extensions (including external monitoring, punishment, and peer monitoring with several kinds of incentives, all appearing in different combinations in the other panels). We then worked through the list of additional mechanisms, scanning single parameters for how they changed the base game's equilibria. Through this process we found that no single mechanism could

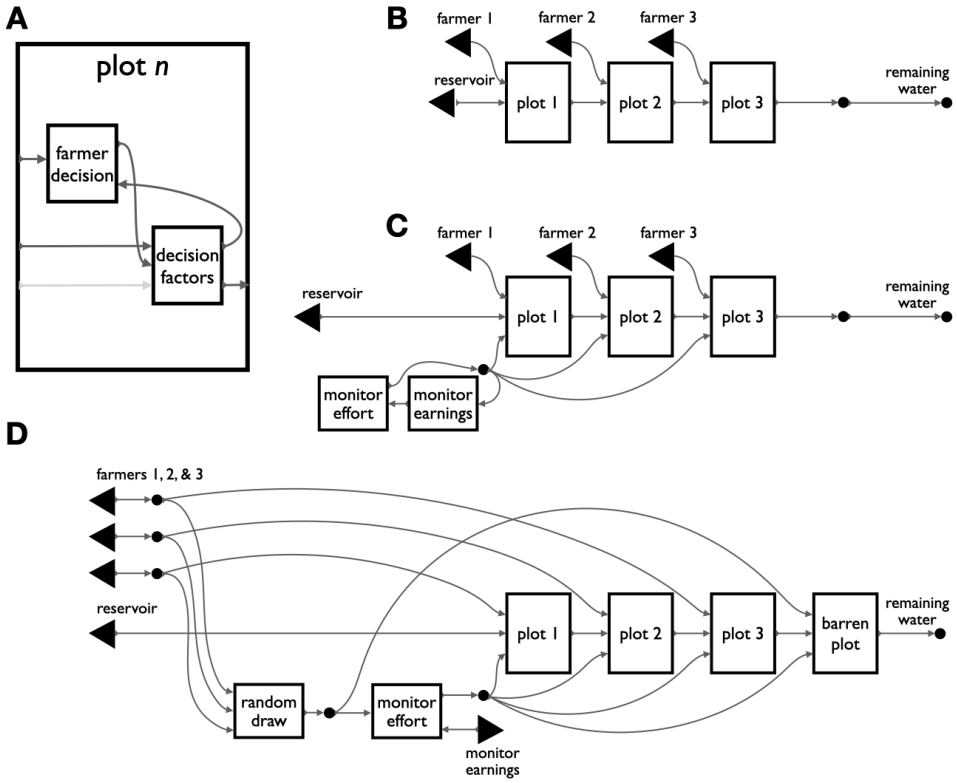



**Figure 2.3.1: Several variations of an irrigation institution.** Compositional game theory brings modeling attention to high-level features such as game structure and evolution. In a sustainability application, it can capture the diversity of solutions to the asymmetric social dilemmas typical of rural irrigation systems. The variants here are drawn from a comparative study of water sharing institutions in Nepal. **A**. This module of game structure represents the internals of each plot of the other panels. Farmers decide how much water to extract for their plot on the basis of their incentives, which are calculated from many different inputs. In a direct analogy to function declarations in many programming languages which define the types of the inputs and outputs, this module can be seen as a function, its inputs are a player string, a water level parameter, and an optional penalty parameter, and its single output is an updated water level parameter. This module's reuse in the other games, with different players entered in different ways, and water levels output from prior calls being used as the input to subsequent calls, all reflect the substance of the mapping from software design to institution design that compositional game theory permits. **B**. In the simplest institution, upstream "headender" farmers extract water from a finite reservoir without concern for the water needs of "tailenders." At typically low reservoirs, no water remains for lower plots. **C**. With minor modifications to B, an external monitor earns a fixed rate to enforce sustainable extraction with penalties. This variant allows the monitor's action to influence the farmer's decision by using the optional third input in the plot module (panel A). **D**. In this sophisticated variant, the farmers rotate through the monitor role, who is incentivized to work honestly with access rights to a fourth field that only receives sufficient water when all upstream farmers are compliant. Compositional game theory facilitates high-level comparison across cases, and iteration through them.

succeed in stabilizing an outcome of equitable cooperation (and that defection by the head-ender is the most common pure strategy equilibrium). Of course, this conclusion holds only for a single iteration of a scenario that, for farmers, repeats every season. With serial composition, the scenario can be extended through time, and represent arbitrary structural complexity.

Note that the diagrams of Fig. 2.3.1 all translate directly into formal computational descriptions; this is a nice consequence of the fact that the category of open games has a formal diagrammatic language [11]. In practice, we generally build the diagrams first to check our understanding, convert those diagrams into code (currently this conversion is done by hand, though it is relatively simple), and then test for equilibria by plugging different strategies into the open game engine [50].

## 2.4 Case 3: Policy development in the Hurwicz model of institutions

*In this example, we consider a simple principal agent game sourced directly from Leonid Hurwicz, and use it to illustrate how compositional game theory can model relatively sophisticated games in a clean way—including games where individuals must choose between multiple games. This particular examples emphasizes the use of the branching operator."*

Mechanism design is the area of economics, and social science generally, most concerned with the formal design and engineering of institutional processes. It has been especially successful in simple or narrowly defined applications, and is increasingly common in political science, for models of the policy design process. For instance, a theme that has occupied decades of interest in political science and political economy is that the process by which a policy is developed can have a dramatic effect on its form [51–54]. This phenomenon requires integrated models of the design, adoption, and functioning of a policy.

With an eye to this problem, Leonid Hurwicz, one of the pioneers of mechanism design, introduced a definition of institutions in terms of families of game forms, and applied the construct to model policy processes [14]. Hurwicz imagines a

classical principal-agent scenario, in which the incentives facing a landlord and sharecropper are determined by their land tenure arrangement (sharecropping, wage-labor, renting, etc.), which is itself determined by two earlier game stages that establish the parameters of the final game. In developing the formalism, Hurwicz proposes abstracting payoffs out of the process model, making payoffs just one kind of game outcome, and including inputs that represent the game's external environment (the "open" in "open games"). Compositional game theory leverages advances in theoretical computer science to offer an abstract, modular basis for Hurwicz's approach. Within a compositional approach, a designer can specify a policy selection process that leaves the specific policy as a black box. Any set of policies with matching type can be substituted in as a module.

We implement a compositional rendition of the Hurwicz landlord scenario in Fig. 2.4.1, in which an employing principal and employed agent interact under a land tenure arrangement selected by an outside decision maker. By overcoming the representational limits of existing game forms, compositional game theory consummates Hurwicz's vision for a generalized game theoretic approach to policy processes. In the process it extends the range of institutions that can be expressed as complexes of games.

## 2.5 Discussion

Compositional game theory is a powerful formalism introducing modularity, abstraction, and expressive power, extensions that make it invaluable for advancing the science of complex institutions. The need for compositional game theory rests in part on the increasing need for a "high-level" game theory, interested in richer and more facile game representations, and "descriptive" game theory, focused less on the formal solutions and solvability of games and more upon expressing the institutional variety observable in empirical case, ethnographic, and historical work in all disciplines.

We contribute a flexible type system for defining general and modular games, supporting tools such as engines for different classes of game and solution concept, and, overall, a principled extension of game theoretic modeling to systems of games

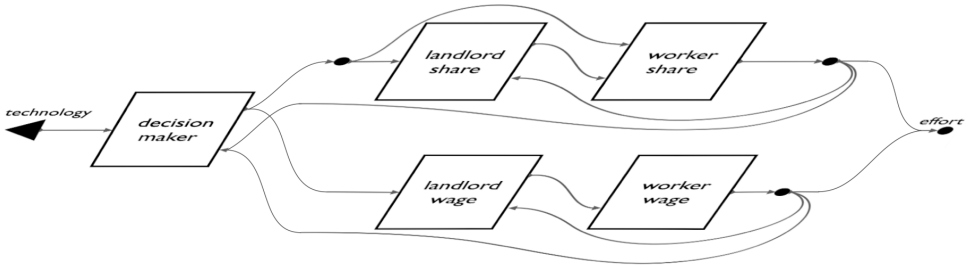



**Figure 2.4.1: A decision maker uses technological information to select between alternate policy designs.** In compositional game theory, a player's actions include choices between games. We use this feature to extend Hurwicz's 1996 model of the policy design process. A modular decision maker component—which could be filled in with a single decision maker or collective process such as voting—computes a preference between different policy approaches to a principal-agent problem, each an open game. The policy chosen will decide whether the agent earns a fixed wage or piece rate. The two regimes lie in different planes because they are mutually exclusive. The decision maker's preference between them is informed by the agent's prospective effort in each regime, and also by incoming information about the outside technological environment. For example, technology may improve the observability of worker effort, which may improve the performance of wages relative to the piece rate. With the regime selected, a landlord and worker play out the selected institution, with workers emitting a final effort.

with complex interlinkages, toward a rigorous computational representation of real-world institutions. This type system is supported by a formal diagrammatic language and a powerful engine [50] for rapidly iterating through games and for computing various solution concepts.

And with three examples of complex institutions in the economic, sustainability, and policy domains, we illustrate the value of extending game theory under the same theoretical umbrella of computer science, which draws representational power from the power of abstraction and composition.

Nevertheless, compositional game theory admits several limitations. Taken one at a time, the operations and design patterns we introduce here all have pre-existing analogues either in the core of game theory or common extensions to it. Behavioral game theorists, for example, commonly introduce election and other choice mechanics in tests of theories of fairness, procedural justice, and cultural evolution [25]. Nevertheless, the framework as a whole clearly goes beyond game theory's core, as even Hurwicz recognized, with his calls for complex trees of game trees and explicitly "non-game" inputs. Our implementation is general, theoretically grounded, and

high-level; it is preferable for designing complex institutions for the same reasons that software engineers prefer modern computer languages to machine code.

Although we assert that solutions and solvability do not account completely for the theoretical power of games, they still have a clearly important place, one that compositional game theory may struggle with. Compared to existing software for finding game solutions [55], our implementation is currently amenable only to a minority of solution concepts. Although it accommodates Nash, subgame perfect, and several varieties of Bayesian equilibria, there are challenges to implementing min-max, Pareto, ESS, perfect Bayesian equilibria, not to mention the theoretical apparatus of cooperative game theory.

Our approach also struggles to decouple equilibrium concepts and structural representations as cleanly as the ideal game engineering approach would permit. In our implementation, the type system that permits game components to be linked is intermingled with the engine that guarantees the solvability of the resulting system of games. This design challenge may be a limit of our implementation, or it may reflect a deeper theoretical link: categorical game theory may require assumptions from the solution concept in order to provide the type and computability guarantees of a rigorous computational framework.

# 3

# Clause2Game: modeling contract clauses with composable games

How do we build game-theoretic models of whole contracts, and what are these models good for? The machinery and use-cases developed in Chapter 2, while more realistic than previous game-theoretic models of institutions, are still too pedagogical and academic for practical use; they are largely aimed at illustrating the viability of open games to a set of social scientists. In this chapter, we refine the tooling and apply it to a particular class of institutions: contracts. In doing so, we work with actual contracts as well as data from a legal tech company, emphasizing a set of real-world, practical use-cases in contract drafting.

Substantial work in contract theory applies game theory to analyze the design of contracts, but contract theory does not typically model whole contracts composed of many clauses due to the practical limitations of hand-built game-theoretic models, nor does it address the institutional diversity found in real-life contracts in the way that, for example, computational law does. In this chapter, we present a practical demonstration of how open games can be used to model actual, real-life contracts in their entirety, as well as a software tool, Clause2Game, to facilitate that modeling workflow. The core of the tool is a new data set of clause models that can be composed to generate games in the same way that clauses can be composed to generate contracts.

In presenting the tool and data set, we emphasize a modular, software engineering approach to contract analysis and design, and discuss the opportunities and obstacles to introducing more ideas from software engineering to both contract modeling and the practice of contract law. This conceptual approach will be important later, as it grounds and connects the work in both Chapter 5 and Chapter 6 on digital agree-

ments and smart contracts, respectively. In particular, this chapter's description of contract engineering gives the fullest description yet of the vision of this thesis: a practical discipline for engineering social institutions.

## Contents



# 3.1 Introduction

Contracts are enforceable promises. Behind the text, the handshake, or the transaction, contracts manifest the intentions of the parties involved, and the consequences when these intentions are not met. Contracts are, therefore, fundamentally relational; the will of the parties is a central pillar to the formation of the contract. In their simplest form, contracts require an offer and acceptance.

Contractual language has since evolved into its entirely own genre of literature. The texts of contracts are rich and complex, a perceivably "viscous sea of verbiage" [56]. Yet, what is contained in these texts are, in fact, the implicit preferences of each contractual party, wrapped in natural language. Contracts, then, represent, in part, the goals and risks parties are willing to bear. Consequently, prior

literature in law and economics has framed contractual exchanges through the lens of economic effects [57].

But how are such economic aspects of contracts analyzed in practice? How can a lawyer systematically understand, for instance, the incentive effects of a subcontracting clause on the conduct of their client's contract partners? The framework we offer, Clause2Game, is a software tool for modeling contracts and contract clauses. The focus of the tool is on strategic interactions and, thus, is particularly well-suited for investigating the incentive effects of contractual arrangements. Once a contract or contract clause is represented as a strategic interaction in our tool, one can explore the consequences of varying specific aspects of the contract. Along with the tool, we also present a data set of pre-built clause models that can be used to explore a number of contract types, from employment contracts to liability schemes to complex M&A deals.

Game theory is and has been a useful metaphor in law [5]. In 1990, contracts scholar Avery Katz discussed the use of game theory for the analysis of contract formation. He suggests that game theory is capable of answering two key legal questions: (1) which actions and intentions suffice to form a contractual obligation; and (2) how do these actions and intentions affect the content of the contract formed [5, p. 216]? Katz argues that while prior literature in contract law has dwelled on consequences following formation, there is merit to considering party incentives during negotiations and how they are affected by various "interpretative regimes." [5, p. 217]. That is, the possible choices available to parties in a prospective contractual exchange, the extent of information available, and their respective costs and benefits to each party should be forthrightly specified [5, p. 223]. This means that the laws of formation must account for the laws of bargaining, such that the parties are made aware of the strategic structures when engaging in contract negotiation.

While Katz puts forth a fascinating theoretical framework, game theory falls apart quickly when one wrestles with more complex, multi-dimensional contracts. Every game-theoretic model has to be painstakingly built by a highly-trained economist, usually paired with a lawyer specialized in the subject being modeled.

Thus, microeconomic analyses of law typically do not model whole contracts due to the practical limitations of these hand-built models, nor do such analyses match up to the institutional diversity of real-life contracts in the way that, for example, more data-driven forms of computational law do. To make game-theoretic models more practical and scalable, we need to import techniques and ideas from software engineering. The goal of this paper, then, is to synthesize software engineering and game-theoretic models in a software application that is useful for contract drafting.

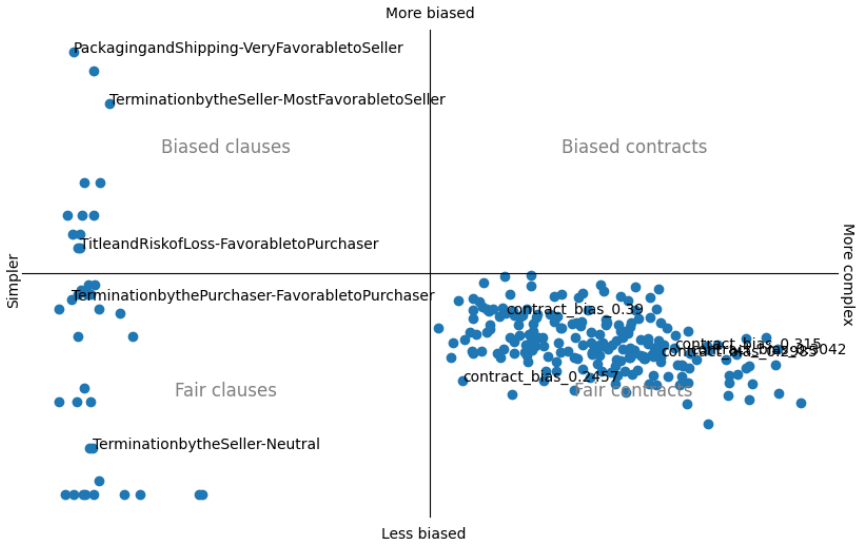



**Figure 3.1.1:** Exploring the contract space of a typical sale of goods contract. In the figure above, each point corresponds to either (1) a game-theoretic model of a potential clause in a sales of goods contract or (2) a game-theoretic model of a composition of clauses, especially those that form a complete sale of good contracts. We then compare these models' strategic properties, including their syntactic complexity (measured by the number of words in their textual form) and their degree of bias (measured by the average difference in payoffs between the seller and the buyer of the goods, across all equilibria). Note that while we have included every single clause model, we have only included a small portion of the nearly one million possible complete contract models. Each model is written in the Clause2Game domain-specific language.

In Part I, we introduce the notion of modularity, review existing studies of modularity and boilerplate in legal contracts, and relate that work to other work in computational law. In Part II, we introduce the Clause2Game software tool and describe the specific software engineering practices that it implements within contract modeling. We also describe how to access the tool and its related data set of clause models. Finally, in Part III, we describe how Clause2Game can be used

within a hypothetical practice that combines elements of software engineering with contract drafting.

## 3.2 Part I: Modularity in law and in software

Modularity is a method for managing complexity, and one obvious way to manage a complex contract is to break the contract down into smaller, more manageable pieces. For example, Henry E. Smith observes how contract clauses, especially boilerplate clauses that are portable between contracts, take advantage of modular design principles, and draws an analogy between writing modular contracts and writing modular software programs [58]. By deconstructing contracts into smaller components (clauses), the complexity of the agreement as a whole can be reduced to its individual promises, making the contract drafting process more understandable and reproducible, reducing the amount of (costly) communication necessary between lawyers and the parties to the contract, and even changing the dynamics of negotiation. Smith also observes how modular principles have been overlooked in traditional law and economics due to (1) the anticonceptualism and antiformalism of legal realists such as Coase and (2) the fact that issues such as information overload are usually not considered in legal and economic theories (notwithstanding economic theories such as Herbert Simon's notion of bounded rationality) [58, p. 1178]. A related commentary by Radin point outs the rise of modularity through three eras of boilerplate grounded in three legal technologies: the era of literal, metal boilerplate produced by a linotype machine, the era of form files to be edited through photocopying and manual cut-and-paste, and the era of computer-aided reproduction and recombination where "modularity now comes into play much more regularly and with much finer granularity." [59]

### 3.2.1 Law, computational law, and game theory

In "The Legacy of Hammurabi," Michael Genesereth highlights applications of logic programming to legal texts which, in effect, require that legal knowledge be made explicit in order for texts to be represented in symbolic form [60]. This use of logic

programming suggests that law, to an extent, is built on logic and that thus logic can be used to investigate and peel back, layer-by-layer, the assumptions of legal concept formation. Similarly, Jones et al. have applied ideas from functional programming to formalize the underlying, modular organization of financial and insurance contracts and to compose those modules into whole contract models [61]. The Clause2Game tool itself is written in a functional programming language, Haskell, and takes advantage of similar ideas, particularly combinators, from functional programming.

The two applications of computational law described above suggest a kind of legal positivism; in crafting and committing to an ontology of legal concepts, we see traces of H.L.A. Hart and his infamous "empirical" system of rules [62]. But not all forms of computational formalization require a positivist commitment. For example, agile methods in software engineering, which we cover in Part II, emphasize a fluid engineering response to the uncertainty that comes from trying to deploy software systems into complex social situations. Compared to traditional, positivist approaches to software engineering based on formal plans and specifications, agile methodologies emphasize hot-swapping and on-the-fly adjustments in response to user feedback. Such methodologies are both more system-theoretic (in emphasizing the interactions between modular, interconnected systems) and information-theoretic (in emphasizing pragmatic elements like bug tracking, error propagation, and information overload).

### 3.2.2 Treatments of complexity

Modularity is a method for managing complexity, but complexity exists at different levels. Law and computational law deal with the complexity of words; logical propositions and functional semantics in computational law can be understood as ways of modeling the complex semantics that arise when one combines words into texts. By comparison, smart contracts as proposed by Nick Szabo and as seen in blockchains such as Ethereum deal with the complexity of code [63]. For example, discussions around modular design in smart contracts center around software engineering concerns like application development, contract calls, and

contract proxies. Game theory, especially as it is presented in this paper, presents a medium between logical representations of text a la computational law and software representations of market transactions a la specific smart contract languages. Games, as strategic models, are high-level representations of the underlying structure and incentives of a contract, and do not represent the text of a contract directly. Indeed, many low-level rules are quite difficult to represent in game theory, which tends to focus on information flow and payoffs. The Clause2Game tool is based on compositional game theory as presented in Chapter 2, a new mathematical foundation for game theory originating in theoretical computer science which has been used to model both smart contract behavior as well as looser institutional structures in political science [11].

## 3.3 Part II: Contract modeling as software engineering

In this section, we present Clause2Game, a software library for modeling legal clauses and contracts. In doing so, we demonstrate how the software can be used to:

1. *model* contract clauses and clause templates,
2. *compose* clause models to form contract models, akin to how lawyers compose clauses to form contracts,
3. *automate* certain kinds of analysis, especially equilibria checking,
4. *search* through and evaluate “spaces" of similar contracts,
5. *connect* contracts to other applications, including other contracts,
6. and enable more *agile* contract drafting, especially in response to greater contract complexity.

Each of the features listed above leverages a particular software engineering practice (see Fig. 3.3.1). In other words, Clause2Game leverages the fact that modeling a legal contract of even moderate complexity is often *already* a form of

programming and thus could benefit from some software engineering to structure that program. In the sections that follow, we make this translational strategy explicit by first describing a practice of modern software engineering and then showing how Clause2Game implements that practice within contract theory.

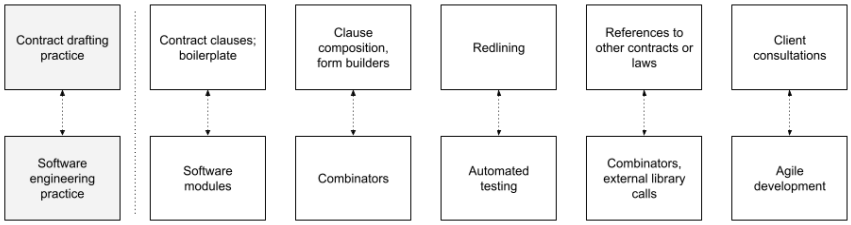


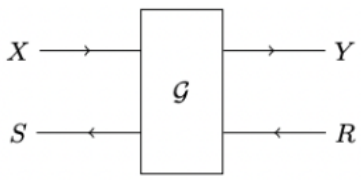

**Figure 3.3.1:** A comparison of standard contract drafting practices with software engineering practices.

### 3.3.1 Modeling clauses

In software engineering, a single program is often separated and decomposed into a series of modules or libraries. These modules usually provide distinct functionalities—e.g. in a banking application, one module handles the user accounts, while another module manages secure login, and yet another computes statistics for the user. These modules communicate and reference each other in order to generate the behavior of the whole program, and they can relate to each other in rather complex ways; a whole class of software, called package managers, has been invented to deal with these complexities. Importantly, software modules are designed to communicate with each other through constrained interfaces. This means that (1) modules help hide and manage complexity behind their interfaces, (2) changes to the code behind a module's interface will not break the behavior of other modules, and (3) like clauses, modules can be reused or "called" in many different places.

Clause2Game organizes individual clause models into software modules called "open games" which can interact with each other through highly-constrained interfaces (see Fig. 3.3.2).

Designing clause models in this way introduces some additional complexity in the beginning, just as building a piece of software in modular pieces takes




```
_NameOfGame_ _var1_ _var2_ ... _varN_ = [opengame|

    inputs    : _InputGame_ ;                 -- This corresponds to X in the diagram
    feedback  : _FeedbackGame_ ;              -- This corresponds to S in the diagram
    :----------------------;

    INTERNALS OF THE GAME

    :----------------------;
    output    : _OutputToOutside_ ;           -- This corresponds to Y in the diagram
    returns   : _ResultsReceivedFromOutside_ ;  -- This corresponds to R in the diagram

|]
```

**Figure 3.3.2:** An open game $G$ along with its interfaces, represented as in (left) wiring diagram form and (right) code. In this case, the game $G$ has an output variable $Y$ (representing the possible actions or moves of an agent in the game), an input variable $X$ (representing the information about actions from other open games), a return variable $R$ (representing the utility that the agent receives from performing actions), and a feedback variable $S$ (representing utility that this game generates for previous games).

some upfront commitment to an architecture. But over time, the benefits can be substantial. Modeling and analyzing complex contracts becomes easier when one is able to combine existing components.

### Example: a typical sale of goods contract

To illustrate the claims in this paper, we describe an extended contract model of a typical sale of goods contract sourced from Lawgood, a widely-available platform for drafting contracts [64]. For example, Fig. 3.3.3 below depicts a small model of two potential clauses governing the delivery location—simply, whether the goods in question will be delivered to the shipper's location or to the purchaser's location. This translates into a relatively trivial game-theoretic representation: a (one-directional) cost function that takes one parameter, the location, and outputs two simple payoff signals to the seller and buyer, respectively (see Fig. 3.3.3). Many clauses can be modeled as these sorts of simple cost functions—and it is often productive to first model a clause as a cost function before "zooming in" to define more complex interactions.

We have reproduced the text of the clauses in Table 3.3.1. As one can tell, many clauses are also characterized by additional potential for internal variation at a set of named subclauses or parameters.

Now consider a different example with more interesting dynamics: inspection. In one option, "Favorable to Purchaser", the purchaser may reject non-conforming goods, which the seller is responsible for replacing or refunding. In the other


**Figure 3.3.3:** A wiring diagram representation of the clause model for delivery location, modeled as a simple cost function that, given a location, outputs a cost for the seller and a cost for the buyer.

| **Option A: Shipper's Location** | **Option B: Purchaser's Location** |
|---|---|
| The Seller shall deliver the Goods to [SELLER'S LOCATION] (the "Delivery Location"). The Seller will send notification to the Purchaser of the delivery of the Goods to the Delivery Location in a commercially reasonable time. [The Purchaser shall take possession of the Goods within NUMBER days of such notification.][Such notification will list the dates and times at which the Purchaser may take possession of the Goods.][The Purchaser shall give to the Seller three (3) days' notice before the Purchaser desires to take possession of the Goods.] | The Seller shall deliver the Goods to the location designated by the Purchaser (the "Delivery Location"). |

**Table 3.3.1:** Options for the delivery location clause in Lawgood's sale of goods contract.

option, "Favorable to Seller", the purchaser must either accept or reject the goods within a specific period of time, after which the seller determines if the goods are non-conforming and must either replace the goods or refund the price (see Fig. 3.3.4).

Note that in the context of this game, the seller is never incentivized to confirm that a good is non-conforming; they can only lose money. So this single clause model, on its own, cannot explain the strategic relevance of an inspection clause.

A selection of the clauses in the sale of goods contract can be seen in Table 3.3.2 below. The full set of clauses and parameters can be seen in Appendix A.

We will delve into different aspects of this sales of good contract as we cover different features of Clause2Game in the next sections.

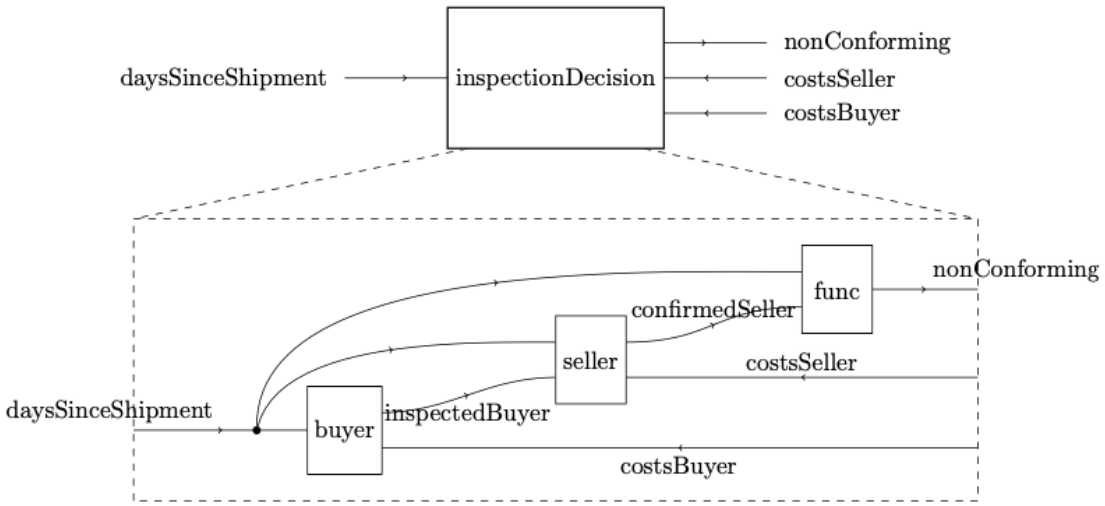



**Figure 3.3.4:** A wiring diagram representation of the clause model for the "Favorable to Seller" inspection clause. In this case, the decision about whether to accept a good depends on decisions by both the buyer and the seller. The seller's decision depends on the prior decision of the buyer.

| **Clause Category** | **Clause Options** |
|---|---|
| Delivery Location | Shipper's Location OR Purchaser's Location |
| Packaging and Shipping | Favorable to Purchaser OR Neutral OR Favorable to Seller OR Very Favorable to Seller |
| Title and Risk of Loss | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Inspection | Favorable to Purchaser OR Favorable to Seller |
| Product Warranties | Favorable to Purchaser OR Neutral (1) OR Neutral (2) OR Favorable to Seller |
| Insurance | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Indemnification | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Limitation of Liability | Neutral OR Favorable to Seller |
| Termination by the Seller | Favorable to Purchaser OR Neutral OR Favorable to Seller OR Most Favorable to Seller |
| Termination by the Purchaser | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Assignment | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Force Majeure | Neutral OR Favorable to Seller |

**Table 3.3.2:** A compositional contract model for a typical sale of goods contract, sourced from Lawgood.

### 3.3.2 Composing clauses

In software engineering and especially in functional programming, functions can be combined using a number of well-defined combinators. Most obviously, one function can completely substitute for another one, supposing that they have inputs and outputs. Two functions can be composed sequentially: the output of one function is taken as the input of another function, forming a new function. Two functions can also be combined “side-by-side", where two or more component functions operate simultaneously but without interacting with each other. Together, these three basic combinatory operations—substitution, sequential composition, and parallel composition—form a rigid but highly-useful calculus for how to build new functions out of old ones.

The software modules or open games of Clause2Game have a similar combinator language, meaning that they can similarly be composed through these kinds of well-defined operations (see Fig. 3.3.5). Perhaps the most important feature of these combinators is that they can help us understand and model the information flow and “strategic flow" in the contract, helping drafters and designers identify which aspects of the contract are relevant in incentive design. Further, because these combinators have strict rules that guarantee their correctness, we can also use them in automated ways. In particular, we can automatically generate entire families of complex contract models from a given set of component clause models, helping us explore the space of all contracts. For exmaple, the Lawgood sale of goods contract framework has 995,328 possible contract variations based just on clause composition, not including additional variations based on external parameters to the contract such as the price of the good (see Fig. 3.1.1). This space of contracts would be impossible to navigate without

Note that in Fig. 3.1.1, the dimensions of the contract space (model complexity and bias) were chosen to highlight the syntactic (model complexity) versus semantic/strategic (bias) properties of these models. Other choices of dimension are possible and interesting, e.g. model complexity in terms of function calls, average welfare across different strategies, and number of equilibria. With sufficient data

(which is harder to capture for some dimensions than others), we can use different features to map families of contracts into equivalence classes; having an equivalence class of contracts with similar equilibria, or similar levels of skewedness, would be especially useful for helping analysts explore the contract space.

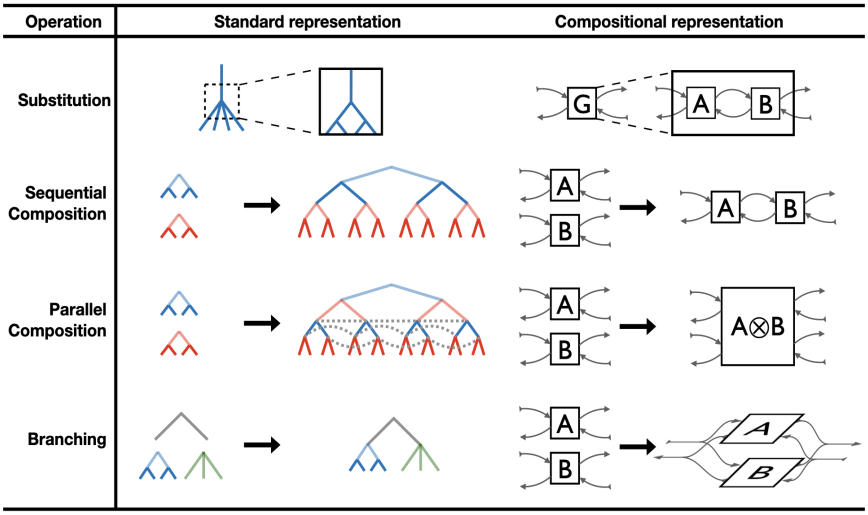



**Figure 3.3.5:** Four operations for combining smaller games into larger games.

To be clear, representations of contracts do not fall from the sky. Each clause model takes time to build and requires detailed knowledge of the domain, and integrating two clause models in any non-standard way (i.e. not using a combinator or combination of combinators) will also require time and detailed domain knowledge. There is no magic which can make this need go away. But in contrast to general simulation frameworks, Clause2Game keeps the needed overhead to a minimum. And, because it emphasizes modularity and reuse, the more people that use it and the more clause models are built, the easier analyzing contracts will become.

### Example: inspection and warranties

Consider the inspection clause depicted earlier in Fig. 3.3.4. Earlier, we noted that the "Favorable to Seller" clause model presents a very skewed game where the seller is never incentivized to confirm that a good is non-conforming and thus be required to replace or refund it. So what is the strategic relevance of including a confirmation stage?

Taking a broader scope, there are many reasons that a seller might wish to confirm that a product is non-conforming, including expectations of future business with the buyer, reputational incentives in the market, laws about product quality and truth-in-advertising, related to promises and/or representations about a given product made outside the sales-of-good contract. These reasons are not obviously encoded or even encodable within the sale of goods contract, though they could potentially be encoded within other games and institutions and models thereof. Some reasons for a seller to confirm, however, are included in or explicitly excluded from the contract, e.g. product warranties and limitation of liability. For example, the "Favorable to Purchaser" inspection clause explicitly invokes the product warranty section: "Except as provided under this Section and Section 4 (Product Warranties), the Purchaser has no right to return Goods purchased under this Agreement to the Seller."

Connecting inspection to warranties to relatively simple when the clause models for each have been built and well-specified. In this case, one simply maps the nonConforming signal in the inspection game to the decision in the product warranty game (see Fig. 3.3.4).

**Example: the YC SAFE**

To give more intuition on what a larger contract model looks like, we preview below in Fig. 3.3.6 a fully-worked and connected model of the YC SAFE, or YCombinator Simple Agreement for Future Equity, a commonly-used contract in the startup industry [65, 66]. The YC SAFE allows startups to take investment easily and safely without setting up a formal shareholder agreement or other administrative costs associated with issuing equity. Despite its relative simplicity, the SAFE contract is still reasonably complicated, with over 20 distinct clauses and multiple variations (e.g. valuation cap and no discount; discount and no valuation cap; most-favored nation with no valuation cap or discount).

There is an immediate practical value to such a model: since open games are code, they can be used to create strategically-equivalent smart contract versions

of the SAFE, versions that allow blockchain-based entities such as decentralized autonomous organizations (see Chapter 6) to take investment while enjoying the guarantees and performance of a SAFE. They can also be used to validate theoretical architectures for blockchain-based SAFEs.[67]

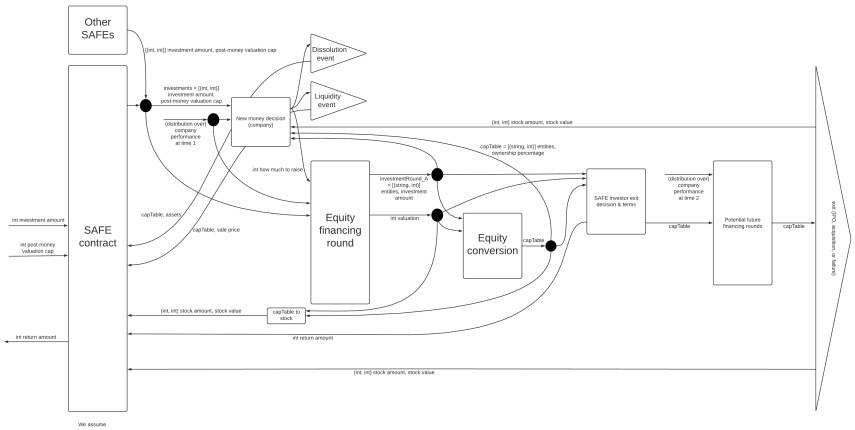



**Figure 3.3.6:** A worked example of an open game for the YC SAFE (valuation cap, no discount).

### 3.3.3 Automating tests

In software engineering, automated tests are used to check the validity and safety of software. These tests are important in the maintenance of complex software as a check on the proliferation of bugs or errors in code. Automated tests enable advanced practices such as continuous integration, where a team of developers regularly merge their code edits into a central repository, sometimes multiple times a day.

Clause2Game ships with several basic automated tests and statistics that can be run on any completed clause model. Most importantly, for any given clause model, we can automatically check its Nash equilibria (among other equilibria like Bayesian Nash) in order to check for problematic loopholes, misaligned incentives, and stark biases. These tests can be combined with many other ways of querying the contract model, e.g. modeling distributions of beliefs or using a random distribution of inputs to detect good and bad strategies.

Catching loopholes is important. For some very simple contracts the examples can seem contrived, but imagine even a moderately-complicated procurement

contract, one that contains a concrete allocation procedure akin to an auction. It is well known in the auction literature that details in the design matter. Equipped with Clause2Game, one can build a concrete model of the allocation procedure and analyze it. If designed from the perspective of the seller, is this format likely to incentivize competitive bidding? Are there aspects which might make the format susceptible to collusion? One can also go one step further in this case. Is it clear that bidders understand the sales process? In that case, one could use Clause2Game as an interactive instruction device so that the contract partners can familiarize themselves with the details and the consequences of their actions.

Further, the economic consequences of standard contracts might be obvious and well-understood. But what if we consider a novel contract with features that are not standard? Or what if the contract is complex? In such a case, there might not be one single economic dimension that is relevant but many. It is at this point that Clause2Game is most promising as it allows us to iterate through many economic interpretations of a contract quickly. It can be used to tap clause models and contracts for weaknesses or problems and to detect boundary conditions in the strategic space, i.e. moments when one action leads to a set of substantially different outcomes.

**Example: limitation of liability**

In order to automate testing, we need to capture more information than the open game clause models—we need to capture enough information to close those games by filling in all the necessary parameters and then to query them for equilibria. This involves some necessary guesswork assigning costs and outcome functions. We are aided in this by Lawgood's data set, which contains favorability evaluations on all the clauses. The results can be seen in Fig. 3.1.1, which capture the outcomes of the model in Fig. 3.3.7.

We describe the full methodology in the appendix.

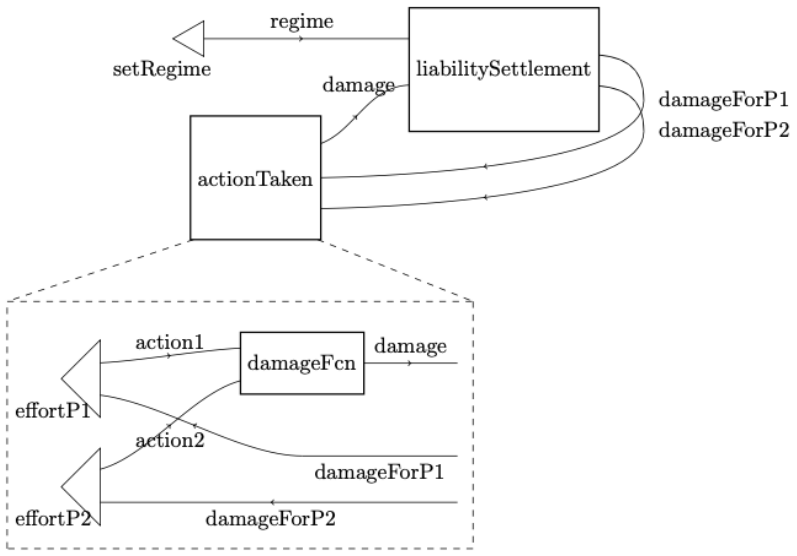



**Figure 3.3.7:** A wiring diagram representation of the clause models for limitation of liability. In this case, a setRegime parameter controls the particular liability.

### 3.3.4 Connecting contracts

In software engineering, modules need to speak to other modules. Or, in modern web architectures, multiple pieces of independent software, living on different web servers, need to come together to form one cohesive application. To do so, these services need to speak to each other through well-defined interfaces called application programming interfaces, or APIs.

Clause2Game is not a web service. But because it defines clear interfaces between contracts, it supports practical integrations with other software applications and even other forms of computational law, from logic programming to newer machine learning models. These integrations are especially important in large, complex contractual arrangements such as merger and acquisition (M&A) deals, where a given contract is often just one component within a much larger decision-making context including analytic workflows, simple accounting models of profit and loss, hugely complicated models that measure the underlying value of a company based on hundreds of inputs, regulatory models that measure the probability of antitrust litigation, and other contracts. Many of these models are already formulated within software; Clause2Game makes it easier for such software to parse and understand

the strategic meaning of a legal contract by providing a well-typed, well-behaved, and well-documented computational model that can be immediately called and inserted into a decision-making workflow.

**Example: Combining payment clauses with actual pricing mechanism**

One standard clause contained in sales-of-goods agreements concerns payments: who receives which payment at which time. Moreover, such clauses often specify what happens if payment is late. In that case, the buyer might be obliged to pay a late-payment fee.

The payment clause exhibits a relatively simple interface to the outside: the actual price, which is the basis for a possible fee, is taken as an exogenous input. While the price is taken as given for the payment clause, in a business context the price is determined somewhere, for instance in bilateral negotiations between seller and buyer. Or, as an alternative, the price could be determined in an auction. When we take the perspective of a seller who wants to simulate the expected consequences for a given sales process, including an underlying contract with specific clauses and including an auction format, we need to reason about the strategic setting and need to understand the consequences of the auction for the wider contract clauses affected by it.

To illustrate the general ease with which we can compose such analyses on different conceptual levels, we have generated a model which takes a specific auction format as given (taken from the baseline open games repository [50], available at `https://github.com/CyberCat-Institute/open-game-engine`) and combined it with the payment clause implemented in Clause2Game. In this way we can derive the possible payment consequences for a pre-specified late-payment fee under the specific assumptions regarding the expected auction revenue.

### 3.3.5 Agile development

In software engineering, agile development (or just agile) refers to an iterative approach to developing software characterized by (1) small, incremental sprints

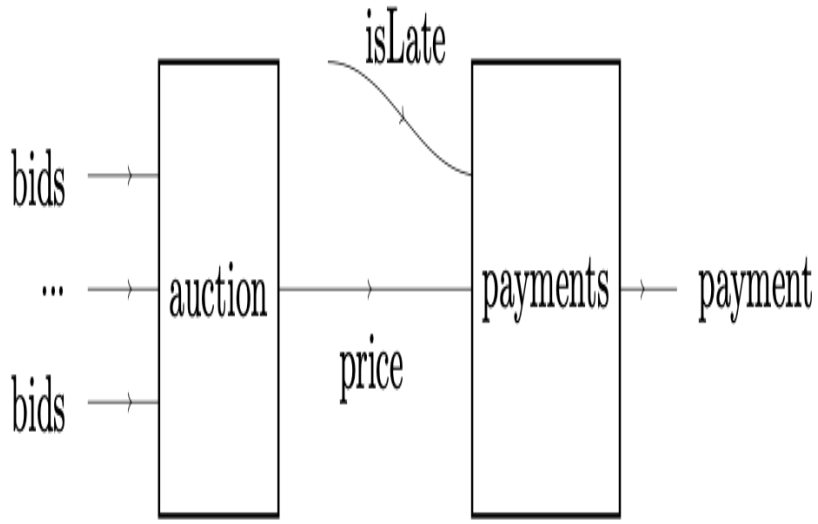



**Figure 3.3.8:** A wiring diagram representation of the clause model for payment, combined with an arbitrary auction game. In this case, a isLate parameter controls whether an additional late fee is added to the payment amount.

organized around minimally-viable prototypes, (2) frequent releases to and feedback from users and customers, and (3) design concepts that facilitate fast, iterative refinement. Agile is based on the recognition that the success of a piece of software depends on the behavior of human users in complex social situations. Compared to traditional, linear approaches to software development that emphasize formal planning over a sequence of discrete stages, agile and related practices "question the assumption that change and uncertainty can be controlled through a high degree of formalization" [68].

Clause2Game can be used in traditional modeling workflows or in agile ones, but the earlier listed design concepts—modularity, compositionality, automated testing—are aligned with agile practices. Almost by definition, contract models are situated within complex social situations; the success of these game-theoretic models (along with the underlying contracts they model) depends on how lawyers, economists, and other parties use and interpret them. And when the underlying contracts change, as they often do in large, complex negotiations, the models need to change with them in order to retain their value.

### 3.3.6 Limitations

To evaluate an open game's equilibria one needs to "close" it, typically with an outcome function that feeds back a concrete utility or other signal in response to players' actions in the game.[1] In a complex game, the outcome function may be extremely complicated, or there may be more than one such outcome function for different components of the game. Compositional game theory helps organize the interactions between many component games, but it cannot specify these outcome functions, which in some sense summarize the behavior of the entire environment in response to player actions. That work must still be done by the modeler.

### 3.3.7 The data set

The current Clause2Game data set of game-theoretic clause models is built on top of Lawgood, a clause library that serves as a practice tool for drafting contracts and clauses. Lawgood is organized into a series of clause categories (e.g. limitation of liability, force majeure, intellectual property, etc.) and then, for every clause category, a set of clauses. Importantly, every clause template in Lawgood comes with a "favorability rating" that suggests, relative to other clauses in that category, how favorable it is to the parties represented in the contract. This favorability rating is not only important for the purposes of game-theoretic modeling, since it suggests an (ordinal) utility function that helps us close and compute the clause models (see section above), but it also provides a data set of expert-provided intuitions that can be checked and verified by the Clause2Game library.

Note that in the data set we require that all clause models be closed. This means that every clause model is a complete and well-described game and can be

[1] While convenient, the word "players" here is not technically accurate; an open game does not keep track of players or roles, per se. Instead, the utilities are fed directly back to the games and ultimately to discrete decisions. On the other hand, the open games engine *does* keep track of roles; effectively, the engine appends a global state function that tracks the names of players. These "role-dependent decisions" were a practical innovation we developed in the course of writing this chapter and Chapter 2 in order to accommodate the design requirements of complex many-player institutions. These new decision types neatly mirror Crawford and Ostrom's emphasis on the importance of roles and rules in shaping behavior within institutions [13]; a role-dependent decision can be thought of as a set of institutional rules that define a role and prescribe behavior within that role.

immediately evaluated for equilibria. While this requires more upfront programming of the clause model, it ultimately allows easier parallel composition of clauses and more closely models the typical way in which clause composition in contracts works, where different clauses are (relatively) independent of one another. The component games of clause models are typically "open" in the sense of having open inputs or "dangling wires". Clause categories are also open in the sense that every clause in the clause category also has a set of shared inputs and outputs. In particular, it's useful to compose clause categories, e.g. an arbitrary warranty clause with an arbitrary force majeure clause. To be clear though, clause categories are *not* categories in the mathematical sense, nor are they well-defined open games, so composition of clause categories is not well-defined.

The current Clause2Game data set consists of a series of clause and contract models, i.e. open games, built in the open games Haskell library. The full data set can be found at `https://github.com/thelastjosh/clause2game`, while the code of the clause models themselves can be found at `https://github.com/philipp-zahn/open-games-hs`.

## 3.4 Part III: Contract drafting as software engineering

So far we have described a technical tool, Clause2Game, for constructing clause and contract models using principles of software engineering. But what are the implications for contract drafting? Supposing that clause modeling becomes more feasible in the future, how would such models change the way that lawyers practice contract law as a whole?

Contract drafting today is heavily-dependent on email threads and Microsoft Word. Most law firms have their own proprietary templates used for a variety of contracts. They are understood as a library of legal documents that embody either the "safest" versions of contracts or the contracts that best represent the general position of the law firm.

In this section, we discuss a range of concrete applications of Clause2Game to contract drafting, first to the current contract drafting processes of law firms, and then to analytics and negotiations within standard form contracts. Finally, we introduce contract engineering as a triangulation of practices within contract drafting and software engineering, and walk through a vision of new practices and applications beyond Clause2Game.

### 3.4.1 Facilitating current practices in contract drafting

We imagine three ways in which Clause2Game might facilitate existing practices in contract drafting (see Fig. 3.4.1):

- *For clause drafting and exploration.* We imagine junior associates, in-house counsel, and/or lawyers at small-midsize firms using C2G as a method of verifying the quality of their drafted clauses. This allows them to have a better understanding of risk outcomes associated with the use of certain clauses over others.

- *For automated contract review.* We envision senior associates and/or junior partners using C2G to act as an additional tool for quality assurance (QA), enabling a more efficient, targeted revision process. In this picture, lawyers review procedurally-generated contracts within a human-in-the-loop system, similar to what partners at law firms already do, with the legal interns and junior associates doing the initial drafting.

- *For business and legal operations.* We imagine those in legal operations are sensitive to importing the best infrastructure for the performance of legal tasks. Using C2G could offer them an opportunity to both draft and review contracts in a much more time- and cost-efficient manner.

- *For contract maintenance and prediction.* The data from increased use of C2G could be leveraged for more quantifiable indicators around the features of a “good" contract.

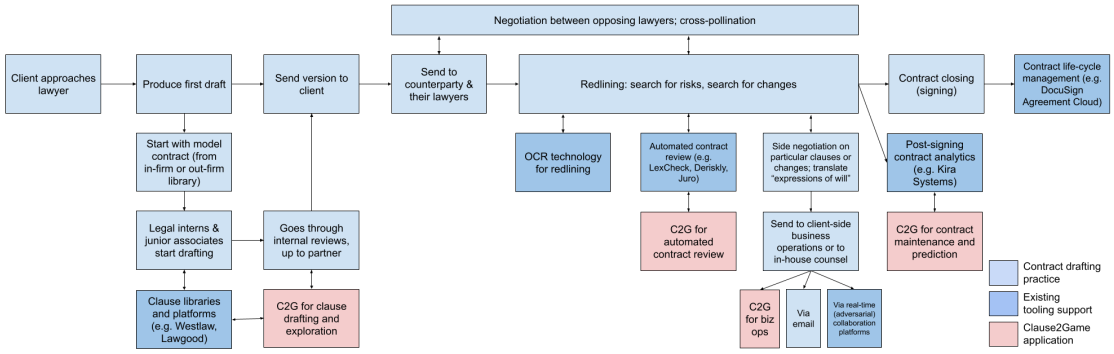



**Figure 3.4.1:** A simple model of the current contract drafting process in a law firm, highlighting places where existing legal tech is used and places where Clause2Game may be useful.

Each of the use-cases above imagines Clause2Game as an add-on service or plugin on top of an existing piece of software, especially software such as Lawgood, LexCheck, or Kira that already consume or produce a structured document model. This aligns with anecdotal evidence we have had in working with practitioners in legal tech (including Lawgood and other companies), who have expressed interest in using Clause2Game's open games model as part of their existing platforms. While it would be harder to integrate game-theoretic constructs directly on top of the unstructured email and Microsoft Word versions of a contract, this may be possible if we can collect enough open game contract data to bootstrap a machine learning pipeline for constructing games from natural language text.

### 3.4.2 Facilitating contract negotiation on standard form contracts

Contract drafting in the broader transactional sense is fast-moving, with details changing at high volumes and speeds across contract iterations. It can be hard to maintain a formal model updated within that workflow, especially when parties can make arbitrary changes to contracts. Alternatively, Clause2Game can be used in conjunction with standard form contracts and/or boilerplate libraries intended for the general public (e.g. typical landlord-tenant agreements, wills). A number of companies (e.g. Willful, Hello Divorce, Rocketlawyer, etc.) offer such standard

language on a freemium model, where the contracts are typically drafted by their own lawyers. We believe that Clause2Game could pair well with such services.

Better analytics may remediate some of the existing power asymmetries involved in deploying these everyday, standard form contracts. While most existing services enable access to “legally-verified" contracts, they frequently do not offer information about the contractual clauses and their respective negotiating positions. Clause2Game can help lay users evaluate the strategic and economic consequences of particular clauses through transparent, quantitative metrics, helping them make more informed decisions without having to consult a lawyer or other specialist.

For example, imagine a renter makes an offer to rent a property from a large, corporate property manager. Almost all rental agreements are drafted by landlords and issued to tenants. Many of these contracts can be predatory, unfair, or illegal. While we do not imagine changing who drafts these agreements, we can imagine certain large landlords, e.g. large apartment complexes, offering some form of automated contract negotiation that allows tenants to trade certain benefits (e.g. pets) for others (e.g. landlord access rights). Renters could apply a points system to trade benefits, where the points system is generated by a Clause2Game model that evaluates the relative costs and benefits of different aspects of the contract.

More generally, Clause2Game might allow lay users to understand their implicit bargaining power *without* requiring an extensive legal background to understand how to read and write contracts. We have seen that readability and opacity of the language is the most cited problem with legalese [69]. However, it may be argued that the readability problem extends further. Simply put, contracts are rather lengthy and on average, most opt not to read them. Even in cases where there are summaries made available, these are frequently considered abstractions of the contract and could run the risk of contractual misinterpretation and misalignment with original party intent. Rather, it may be more effective to enable tools that encourage better signposting and information that directs the everyday individual to gathering specific information about their contracts and the clauses within these contracts [70]. In this way, Clause2Game introduces more agency into

conversations involving standard-form contracts, in turn, empowering the drafting and negotiations of everyday contracts.

### 3.4.3 Contract engineering: one possible future for contract law

What is the payoff to making contract drafting more like software engineering? And what would it look like if we kept pulling that thread? Clause2Game explores some of those trajectories. In this last section, we go beyond Clause2Game to define *contract engineering* as a systematic approach to contract drafting that applies processes, tools, and principles traditionally associated with software engineering. We believe that contract engineering represents a powerful evolution of contract drafting that could leverage existing benefits found in software writing practices and find their relevant counterparts found in classic contract theory and related theories such as sociological jurisprudence.

For example, current contract drafting largely consists of redlining unstructured text documents. But increased use of automated testing within redlining could eventually evolve into a form of automated contract review where contractual analysis becomes not only more focused, but also one that is sensitive to risk allocation and dispute avoidance. One imagines a lawyer opening a legal contract within a "legal code repository" similar to GitHub, and opening a pull request to merge changes. Such platforms are not currently set up to facilitate contract drafting, but they could be adapted to such a purpose. Furthermore, lawyers already organize contracts, through prosaic mechanisms like clauses and numbered sections, and many law firms have existing libraries of clauses and contracts. However, it is hard to treat these clauses and contracts within a truly "plug-and-play" form of contract drafting. Contract engineering could involve defining a standard way of interpreting how a clause relates to another clause, for example by giving the sectional organization of a contract document a more standard and definitive role in defining contract meaning.

Alternatively, lawyers currently consult clients through a long and expensive drafting and negotiation process. But what if contracts could be drafted, shipped,

and signed—and then edited on the fly, as in agile development of software applications? This would require implementing a framework for evolving enforcement regimes (and penalties) through a series of trial periods and maturity levels, as in software development. As in the example above, a framework for agile law would also benefit from platforms for the co-editing, debugging, and integrating changes that allows lawyers to make quick, small changes to a working contract and then to receive some form of quick feedback on those changes (i.e. that it does not contradict some other part of the contract).

More concretely, we consider a potential use case of how contract engineering may be envisioned. Extending the paradigm of standard-form contracts, we consider a space where Terms-of-Service (ToS) agreements may be better evaluated for how their clauses represent the underlying liability structures between companies and their users. This could enable more precise interpretations of the risk allocation across users and respective companies, particularly in the event of an audit. Moreover, Clause2Game could behave as an explainable layer, providing insight into how the clauses reveal the specific strategic positions of the company and what might be potential remedies for users.

## 3.5 Acknowledgements

We would like to thank Nick Stares and Jules Hedges for helpful comments in the preparation of this manuscript. Joshua Tan would like to thank the Mercatus Center at George Mason University for their support.

# 4
# Modular politics

What are the building blocks of governance within online communities, and through what system can we arrange those building blocks? In the first half of this thesis, we sought to scale up existing modes of institutional analysis using tools and techniques inspired by modern programming languages and software engineering practices. Yet, despite this deep indebtedness to code and technology, none of the institutions we have modeled so far are digital-native. In the second half, we pivot to cyberspace, where we can not only build models of social institutions but program the actual institutions insofar as they are embedded in code. The affordances of the online setting are particularly important as they ground a set of bottom-up experiments developed in Chapter 5 and Chapter 6.

In this particular chapter, we propose a conceptual framework for self-governance in *online communities*, emphasizing the particular affordances of online platforms as well as a set of sociotechnical requirements including modularity, interoperability, portability, and expressiveness designed explicitly to foster more bottom-up governance experimentation. This chapter serves as the conceptual twin to Chapter 2 insofar as both chapters present basic theories for how to structure and compose a set of institutional primitives, though Chapter 2 focuses on the development of single (and linked) institutions, while this chapters focused on the architecture and underpinnings of a macroscopic ecosystem—macroeconomics versus microeconomics.

## Contents

## 4.1 Introduction

Many of the pressing social questions surrounding Internet technology and culture are questions of governance. *Who should set policies that govern others' behavior? What content should and shouldn't be allowed? What should be done with users' personal data? How should economic value be distributed?* Yet the governance tools available for online networks—such as social media groups, multiplayer games, and peer-production projects—are remarkably impoverished, typically relying on assigning unchecked power to admins and moderators [71]. Such mechanisms as elections, boards, term limits, and transparent decision-making are norms for any incorporated entity in robust legal regimes, but to the extent that they do occur in online networks, they are costly to build and maintain, and they typically must be implemented by means extraneous to the feature-set of the network's platform itself. Perhaps it is thus not surprising that monarchic or oligarchic practices emerge in online networks more easily and frequently than liberal or democratic ones [71–73].

We propose a strategy for addressing this lapse by modeling and specifying a governance layer for online networks with the following eventual design goals:

1. **Modularity**: Platform operators and community members should have the ability to construct systems by creating, importing, and arranging composable parts together as a coherent whole.

2. **Expressiveness**: The governance layer should be able to implement as wide a range of processes as possible.

3. **Portability**: Governance tools developed for one platform should be portable to another platform for reuse and adaptation.

4. **Interoperability**: Governance systems operating on different platforms and protocols should have the ability to interact with each other, sharing data and influencing each other's processes.

To achieve these, we suggest the development of an open standard, along with supporting software libraries. We call our model "Modular Politics."

The model is preliminary and provisional, a starting point for future work. Modular Politics does not constitute a solution to aforementioned challenges such as content moderation or value distribution. No computational system can capture the embodied and cultural fullness of human governance practices; at best, computational tools can facilitate those non-computational processes. Nor do we argue for the superiority of some governance mechanisms over others. Rather, our proposal points in a direction of how users and platform designers might adjudicate challenges in creative and diverse ways. By proposing a pliable set of tools, we recognize that Modular Politics can allow the implementation of oppressive regimes as well as more democratic ones—but we suspect that pliability will result in greater accountability overall than what emerges from the autocratic administrative tools that prevail on social platforms today. Evaluating this suspicion would become far more possible with a laboratory like Modular Politics at hand.

Despite the near-ubiquity of monarchy and oligarchy on dominant social platforms, diverse tools for governance are as old as the social Internet [74]. These range from the experimental 1982 text-based game Nomic [75] and the VOTEMGR

software available for the early FidoNet bulletin board system [76] to current cloud-based platforms like Election Runner that assist, for example, in elections for corporate boards and "liquid democracy," which several Pirate Parties around the world use to build their political platforms [77]. Platforms such as Loomio [78] and Decidim support deliberation and community building as well as various voting mechanisms. The self-hostable multiplayer game Minecraft allows administrators to install "mod" plugins that can include tools for governance [73]. A new wave of governance systems is emerging among distributed-ledger technologies, particularly surrounding the concept of decentralized autonomous organizations, or DAOs [79–81], which we develop and contribute to substantially in Chapter 6. Yet most users of social platforms do not have the means of experiencing the diverse range of possible governance technologies. Modular Politics seeks to make a wider range of governance possibilities available to users and researchers—possibilities that have yet to be meaningfully tested or even tried.

What follows is a conceptual overview intended as a prologue for future theoretical, technical, and empirical work, not a specification or theory. For example, this overview does not consider matters such as security and database structures that will be vital at the design stage, nor does it present the kinds of empirical claims that we expect to develop and test with an eventual prototype.

First we will review other approaches to designing online governance regimes that motivate our design goals. We will then present a preliminary sketch of the basic features that we expect Modular Politics to include, followed by some examples of how our model might work in practice. Next, we consider strategies for implementation, proposing the development of an open standard, and conclude with a discussion of future research directions enabled by Modular Politics.

## 4.2 Background

The governance of online communities has long been a topic of study within the fields of computer-supported cooperative work, social computing, and computer-mediated communication. Early studies examined discussion groups hosted on

newsgroups [82], mailing lists [83, 84], Internet Relay Chat clients [85], or bulletin board systems [86]. In the past several decades, researchers have gone on to study the governance of large-scale peer production communities such as Wikipedia [87] and open source software projects [88], online multiplayer games such as League of Legends [89], as well as large social media platforms such as Facebook [90].

### 4.2.1 Problems in Online Communities

Participants in early discussion and gaming communities complained about re-occurring issues of unwanted behavior, including spam and harassment [91], as well as conflicts with other members, sometimes called "flame wars" [92]. Today, online harassment and trolling have gone mainstream with the ubiquity of social interaction online, resulting in evidence that nearly half of Internet users in the U.S. [93, 94] and nearly three quarters of people who play online games have experienced some form of online abuse [95].

During the 1990s, software peer production communities [96] began to form, with famous examples such as the GNU/Linux operating system [88, 97] and the collaborative encyclopedia Wikipedia [87]. Like discussion communities, peer production communities must contend with unwanted behavior such as vandalism [98] and sockpuppetry [99]. However, peer production communities often maintain stricter standards for contribution. This can result in barriers for newcomers, including having their first contribution met with no response, impolite responses, or responses that are too complex for them to understand [100]. Peer production communities also may have certain productivity goals, and repetitive conflicts such as "edit wars" on Wikipedia [101] or other intransigent disputes [102] can reduce productivity.

Communities face additional governance challenges as they begin to grow in size [103, 104]. In the case of discussion groups, which tend to have only one or a small number of moderators, those moderators complain of time-consuming labor needed to filter out content [84], as well as burnout from the emotional toll of constantly encountering toxicity [105]. To manage an increasing volume of content, some platforms have invested in machine learning tools to automatically remove unwanted

content [106]. While spam today is mostly dealt with automatically, mitigating other forms of unwelcome content such as misinformation and harassment is harder to automate because of their contextual nature [107, 108]. With a growing number of community members, many may be unaware of established norms [109]. While some communities resolve these problems with more rules governing participation [110], they also run the risk of turning away newer members [100, 104, 111], resulting in low conversion rates of casual users to core contributors [112] and entrenched leadership [113].

The problem of scale reaches new magnitudes when it comes to massive, centralized social media platforms, such as Facebook or Twitter, where much of online social activity resides today. Governance decisions by major platforms, such as the early decision by Facebook to enforce a real-name policy [114], have tremendous ramifications for society given the size of their user base and the inability for many people to opt out of such platforms to participate in society. As they have grown, platforms have become the "new governors" of an increasingly digital public sphere [90], and the governance that platforms provide has become a major component of their product offering [115]. In that regard, platforms have been criticized for their lack of transparency around governing [116], the slow response to emerging user problems such as harassment [117], the outsourcing of content moderation to large teams of contractors facing poor working conditions [118], and the lack of user input into governance decisions beyond low-level flagging [119]. Along multiple dimensions, the governance mechanisms currently available do not appear up to the task of meeting the pressing problems that social platforms face.

### 4.2.2 Governance Approaches in Online Communities

Recognizing the importance and ubiquity of these problems, researchers have taken interest in how communities tackle them. The findings from such work have informed our approach in what follows.

The largest social media platforms have attracted significant attention for their responses to governance problems, with a particular focus on the broader societal

impacts of their governance choices [115, 118, 120–124]. The literature tends to take a critical posture, highlighting the marginalization of user input [119, 125, 126] and focusing on how their profit motive results in governance decisions that are not in the interests of users [127, 128]. Analyses of centrally governed communities become more sympathetic as those communities become smaller and flatter; for instance, the group of researchers that produced *Building Successful Online Communities* [129] provide a well-organized overview of insights into the problems these communities face and strategies for solving them.

Closer to the democratic ideals of early Internet evangelists [130], self-governing communities occupy several seminal studies of online culture, as in research on the LambdaMOO and MicroMUSE platforms [91, 131, 132], in which crisis events galvanized virtual communities in ways that spurred action and even led to formal governance schemes. Several important online organizations employ remarkably democratic governance models. These include the World Wide Web Consortium, the Wikimedia Foundation, the Apache Software Foundation, the Debian operating system, and, after a recent restructuring, the developers of the Python programming language. Among these large-scale, more-or-less participatory organizations, the projects of the Wikimedia Foundation have attracted the most governance research, English Wikipedia in particular [87, 110, 133, 134]. Other large platforms have dabbled with more circumscribed experiments in democracy [89, 135], or seen such experiments emerge from the user community [98, 136–138]. While these more democratic examples play crucial roles in Internet infrastructure, their governance is more an exception than the rule. They also tend to rely on bespoke processes and tools, rather than easily replicable ones.

Alongside such procedural democracies, researchers have investigated the many platforms that rely on more decentralized structures. Examples of this include USENET [82], mailing lists, Facebook groups, Reddit communities, file-sharing networks [139, 140], the Tor network [141], blockchain protocols [80, 81, 142], and, of course, the World Wide Web itself [143]. Independent units of these platforms may be seen to adopt many types of governance styles, but considered as populations of

populations, they can be characterized by the freedom users have to exit any one community for any other—that is, they favor "exit" over "voice" [144]. Member communities of these platforms are most likely not represented in platform-scale decisions, but through grassroots collective action, campaigns have exercised the platform-scale influence they formally lack [145, 146].

This self-organization is especially evident in their ability to develop free and flexible "off-platform" software tools for addressing the governance challenges they share in common, whether via general tool libraries [147, 148], templating mechanics [110, 149], or even community-led randomized experiments [150]. For example, in lieu of centralized platform action on unwanted behavior, community members and end users have piloted new governance designs to dissuade norm violators [151] and protect against harassers [117, 152], as well as to facilitate reconciliation between conflicted members [153] and provide community support for victims [154]. Self-organization is also evident in the ability of communities to develop and sustain widely held informal norm systems [155–158]. Research on online criminal communities is an especially rich source of insight into the potential of norm-heavy governance systems [141, 159]. Complementing the research community's overall faith that technology *can* play a supporting role in online governance, researchers agree that technology alone does not define a governance system or determine its success. Cases abound demonstrating the importance of a healthy culture to a platform's continued existence, regardless of its governance technologies [109, 160–165].

Prior research suggests that although sophisticated and democratic governance is possible in online communities, it is difficult and rare. The tools available to communities far more often leave users to develop governance practices in an informal, *ad hoc* fashion. While we cannot claim that improved governance tooling alone would solve the vexing problems of social platforms, there is reason to believe that an improved framework for experimentation with governance models could generate pathways for improvement.

### 4.2.3 Institutional Analysis and Development

Questions of governance design are of great interest outside of the sciences of information and technology, particularly in the social sciences. Yet social scientists have generally been reluctant to move beyond classification and characterization into design. Plato and Aristotle each offered taxonomies of political systems, employing such labels as monarchy, oligarchy, and democracy; their emphasis on categorizing macro-level structures—assuming a closed set of possible governance forms—continues to hold sway, even as more recent typologies of comparative politics introduce multi-dimensional complexity so as to reflect the interplay of diverse institutions and practices found in modern governments [166–168]. Quantitative comparative political scientists have a much finer-grained view of governance design, but still restrict their focus largely to national-scale governance, exploring questions focused on political organization [169, 170], political processes [171–173], and political economy [174]. Governance design approaches in other social sciences are open to smaller-scale subjects, but have tended to adopt the convenience of assuming a powerful designer, such as a founder, CEO, or administrator. This describes the dynamical systems [7] and cybernetics [8, 9] branches of complex systems theory, which were both introduced as social design paradigms. Design approaches in the management literature on organizational processes recognize managers as a type of designer [175–177]. Economics has a rich literature on market-mechanism design, which has been successful enough to influence contexts from voting systems to spectrum allocation [178–181]. Alternatively, participatory design frameworks are premised on the importance of broadening the design process to include all types of stakeholders [182, 183]. This participatory tradition has been formative to the development of CSCW and modern HCI [184].

Even allowing this range of examples, we conclude that social science has given surprisingly little attention to general and systematic techniques that a lay practitioner can use to craft governance institutions. Fortunately, growing interest in natural-resource management and skepticism toward rigid market/state dichotomies have led to rapid development in a form of institutional inquiry oriented around

the design of governance across a wide range of social contexts, one beginning with participant agency rather than macro-level structure. This effort has been centered at the Ostrom Workshop at Indiana University, founded by political scientists Elinor and Vincent Ostrom. Their Institutional Analysis and Development (IAD) framework, though focused on analysis, is well suited to the challenge of design. In fact, it was aspirationally intended to become a design tool: in IAD's first instantiation, the "D" stood for "Design" [185].

An important feature of IAD is its sensitivity to the diversity and complexity that tend to emerge among actually existing human governance regimes [186–188]. Many of the cases the Ostroms and their colleagues studied were from among longstanding, Indigenous, and non-Western societies that operated quite differently than the usual subjects of economics and political science research. This commitment to empiricism across diverse settings has put IAD at odds with the theory-driven approaches that have dominated the sciences of governance, a tension captured by "Ostrom's Law," which asserts that "a resource arrangement that works in practice can work in theory" [189]. IAD's empirical orientation, and its consequent openness to not-yet-theorized governance forms, has equipped it to capture institutional complexity. Empiricism also makes IAD suitable for a more hands-on approach to institution design, a divergence from the tendency to privilege theory in the governance analyses of economics. This commitment to the experiential component of design was important to the Ostroms, who emphasized the craft of institution design and likened its practitioners to artisans [190].

There are, of course, shortcomings to applying IAD to online governance. Although some have argued that IAD is basically aligned with contemporary critical discourse around power and culture [191], a credible case can be made that IAD's commitments to rational choice and methodological individualism—consequences of its alignment with quantitative behavioral science—make it unprepared to adopt "a broader concept of human agency, contextualise and emphasise the importance of history, and look at the social construction of rationality, interests and identity" [192]. IAD has also not yet been adapted into a computational governance environment

at the scale we propose. Nevertheless, we find IAD to be a promising basis for modeling agency in a computational context.

IAD's basic unit of institutional structure, the "action situation," is a game-like confluence of conditions in which participants make individual choices and collective decisions [193, 194]. IAD normally allows for participant involvement in shaping the core rule-set of an action situation in their shared institutions [195]. To represent more complex institutions, IAD represents action situations as becoming nested and interlinked into systems of larger action situations that, when combined with a set of users and other context become an "action arena" [186]. The Ostrom Workshop community went on to develop a general concept of polycentricity for effective action arenas, which describes the even more complex interplay of multiple institutions that the Ostroms observed in the field [196, 197]. The polycentric frame can encompass nested, overlapping, and competing divisions of power that interact to shape participants' action situations. For example, how storm water runs in and out of a city may depend on activity from many levels of government (federal, state, county, and municipal), many divisions of government (planning departments, parks departments, and waste management departments), many modalities of government (legislatures, courts, and law enforcement), and many stakeholders in government (officials, NGOs, and citizens). A recurrent finding of IAD research is that governance organizes the complexity of each level to match the complexity of the social-ecological system with which it interfaces [187].

Previous research about online governance has adopted the IAD framework [73, 80, 82, 87, 88, 129, 143, 195, 198–200]. We argue for a further embrace of the IAD approach in online governance design, adopting IAD's emphasis on the agency and creativity of participants. The notion of governance as interlinked action situations is ideal for designing complex governance environments from a user-centric point of view, rather than presuming an idealized and highly constrained agent (as in game theory) or abstracting away participant agency by focusing on structure (as in much comparative political analysis). Still, IAD scholars have yet to rigorously formalize the precise nature of these aggregations and their linkages: what specific

ties are permitted between games, or how they come to faithfully represent a real-world institution [188]. Indeed, the particular contribution of Chapter 2 was to provide a set of concrete tools to formalize these linkages so that they could more faithfully represent real-world institutions. The approach we propose in this chapter emphasizes both the affordances and the missed opportunities of the online context in fostering linkages, especially through the use of (deeply-technical) tools such as open games. In the language of IAD, it could be described as inviting communities to formalize and encode the action situations that constitute their own action arenas, which communities can thread together across polycentric networks.

### 4.2.4 A New Approach

Given the bewildering variety of governance institutions and their complex relationship with culture, an online governance paradigm should enable communities to built nested assemblies of action situations. Doing so requires an approach to governance systems that is *modular*, so that users can compose systems of nested systems from the bottom up, following IAD's nested conception of institutions. Elinor Ostrom conceptualizes this nesting in terms of Arthur Koestler's idea of the "holon," which is when "a *whole* system is part of a system at another level" [186]. The standardization that modularity requires should be in balance with *expressiveness* sufficient to implement highly diverse, culturally heterogeneous systems that IAD's global perspective highlights. Finding this balance may involve compromises that limit expressiveness somewhat to ensure modularity and other design goals.

As communities struggle with changes in their user base or operating environment, they benefit from being able to quickly iterate through targeted institutional changes. This process will be easier if communities are able to borrow and adapt working solutions from each other and engage with in the "institutional artisanship" that the Ostroms imagine [190]. To facilitate this, a modular governance paradigm should also ensure that its components are *portable* across contexts. Successful modularity will facilitate portability by requiring the components of a system to have certain standard features.

Online communities are networked communities. Participants may share hyperlinks with each other from across the Web, or leverage a following on one platform (e.g., on Instagram) to achieve their goals on another platform (e.g., crowdfunding on Kickstarter). If a governance system is to encompass the polycentricity of life online, it must be *interoperable* across platforms, able to interact with far-flung data sources and interactions. Modularity, again, can help facilitate this by ensuring that the components of disparate implementations share common interfaces.

With Modular Politics, we seek to outline a paradigm that empowers online communities to assemble discrete, programmable building-blocks into original and complex governance regimes that reflect their aspirations and needs. The model we describe is capable of implementing governance mechanisms, connecting them into higher level constructions, organizing information, and managing resources. Tools created for one community can be adopted in others. The model also supports interaction among distinct communities and across networked platforms.

## 4.3 The Model

In this section, we present a preliminary overview of how the Modular Politics model might operate, drawing on the four design goals outlined above as well as the IAD research paradigm. We employ a set of specialized terms with meanings specific to this model, whose initial letters we capitalize to distinguish our usage from colloquial meanings. These terms serve to establish a technical vocabulary for the model, but in specific implementations, developers may replace them with a user-facing vocabulary more appropriate to their world-building context.

We present the model alongside a hypothetical use-case to help illustrate how each component might work. Consider a fictional multiplayer virtual game, Guilded Age, in which teams known as guilds compete with each other to acquire resources and earn points by building monuments. The game operates on a federated network structure in which each guild resides on a separate user-administered server across a shared protocol, similar to games such as Minecraft or World of Warcraft. Guilds may be as large as 100 players each, and they make collective decisions about

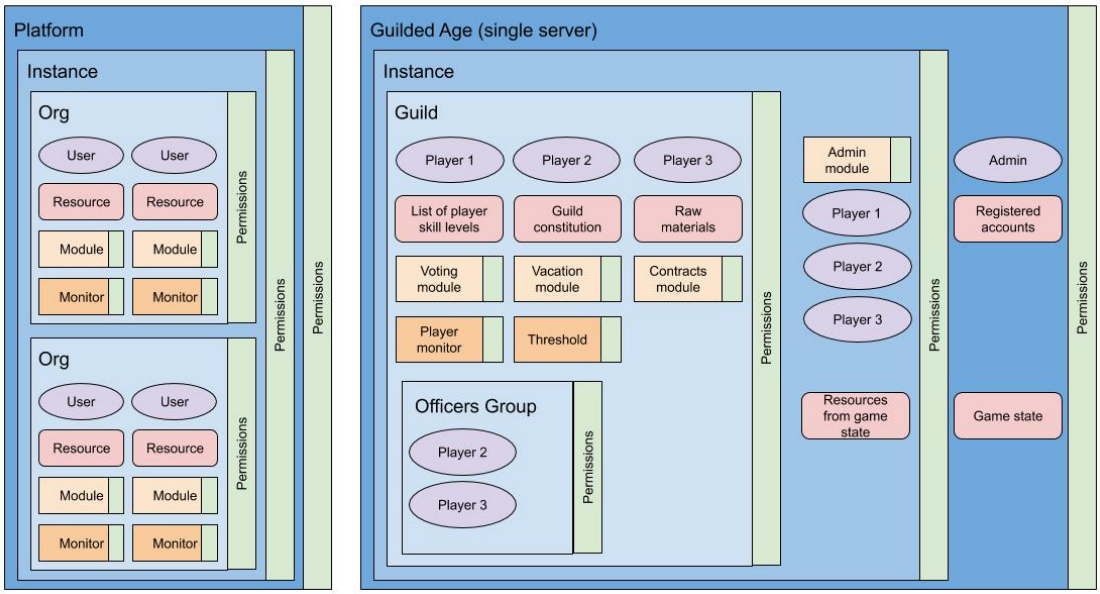



**Figure 4.3.1:** A schematic implementation of the system described throughout this section. Here, a single server running Guilded Age also runs an Instance of Modular Politics. The Instance translates game data into assets recognizable by different Orgs and Modules. Finally, guilds in Guilded Age correspond to Orgs, and each guild may install a number of Modules and track a number of Resources. An Org may have a sub-Org, e.g. a guild may have a sub-group of officers with special permissions.

strategies for resource acquisition, architecture, and construction. The rules of the game are fixed, but the guild associations can choose governance structures for themselves from a large pool of available components. Players are free to choose which guild they want to join for a given round—or they can form their own and attempt to attract others to join. At the start of each round, server administrators assemble a set of governance structures for the guilds they want to form, and other players can use a guild-comparison tool to examine the governance regimes and member attributes in each guild before choosing which guild to join. These guilds can take a variety of forms. Guild founders with high levels of reputation in the game tend to succeed in attracting players to join autocratic guild structures, as past accomplishments imbue trust in those founders' leadership. A diverse range of options for governance thus emerges, as does an ever-growing variety of creative governance techniques.

### 4.3.1 Platforms, Instances, and Orgs

Modular Politics is not a standalone system but must be implemented within an underlying *Platform*—perhaps a game, social network, or blockchain protocol. An implementation of Modular Politics is called an *Instance*. The Instance defines the interface between Modular Politics and the underlying Platform—specifying the entities from the Platform that have a role in the governance process and the actions that they can take on the Platform (Figure 4.3.1). In defining the Instance, Platform operators determine who has access to Modular Politics and what it has the power to govern. The Platform in Guilded Age is the game's software, which runs on user-administrated servers across the network. An Instance of Modular Politics is embedded in the software and thus runs on each server concurrently. The game software defines certain ground-rules for each Instance and associates events in Modular Politics with events in gameplay.

Within an Instance, governance occurs through computational domains called *Orgs*. Orgs provide the institutional context for what IAD calls action arenas in Modular Politics. They are semi-autonomous governance environments, of which an Instance can have many. These are the tools by which Guilded Age players begin to craft the governance of their guilds. Org creation allows for mimicking a variety of organizational types, such as companies, clubs, factions, boards, and more. An Instance's Org or Orgs may be specified by Platform operators in advance; the system's participants may also be allowed to create and join their own Orgs voluntarily. In Guilded Age, server administrators can choose either option for their users. Orgs can be nested or formed in parallel; when one Org is created within another, its members can include any members of the parent Org. Orgs are constrained only by the fact that their members are also subject to any other Orgs within which they are members. A Guilded Age guild can thus have sub-groups within it. This accords with the practice of "nesting," which Elinor Ostrom observed as a recurrent pattern in commons-based governance.

### 4.3.2 Modules

Following the IAD concept of modular action situations, participants in Modular Politics take action through computational *Modules*. Modules are configurable software packages that can alter the behavior of processes that run on an Instance or on particular Orgs within it. A Module's editable configuration options are called *Policies*. Compatible Modules can be combined to form more complex Modules. Modules, as their name suggests, are critical for achieving the design goals of modularity in this model, as well as arbitrary expressiveness; rather than prescribing specific governance systems, Modular Politics enables a bottom-up approach to system creation.

Guilded Age administrators, for instance, might choose to define a certain set of Modules for their players, or they might load a Module to the Instance that provides players with a process to add Modules and Orgs or alter Policies during gameplay. Administrators and players can adopt Modules—or packages of Modules—from third-party repositories or create their own.

The designs for Modules may include typed inputs and outputs, which help facilitate conjoining Modules into ever more complex ones. The inputs and outputs can be Users, Resources, Orgs, or other types of data; inputs and outputs can also come from other Instances (or non-Modular Politics systems) through API calls, if Policies allow. A Module that outputs a decision can provide input for a Module on a separate Instance that translates the decision into a Policy change. In Guilded Age, the guild on one instance might use this method to form a contract with another guild on another server; if one guild violates the contract, the Module automatically pays restitution to the other server's guild. This kind of interoperability is essential for supporting what IAD regards as the inevitable emergence of polycentricity.

Modules are also transparent. Their underlying source code is accessible to anyone who can view and interact with it, so users can audit the mechanics of their behavior. This includes Modules on separate Instances being accessed through API calls; the API enables remote access to Module source, both for inspection and adoption, although a Module's Policies may have restricted permissions. Because

they are all on a common network through APIs, Guilded Age guilds can see which Modules are running on opposing guilds. This buttresses the model's design goals of interoperability and portability.

Modules can have more than one functionality depending on how they are called. For example, a single Module for petitions can be called to either create a new petition, sign an existing one, or query a petition's status. Policy configurations specify these various behaviors. A Module for an elected guild leader will include a Policy specifying the length of the leader's term and another specifying the threshold of support required for the leader to be elected. To operate like this, Modules must be able to interact with, and change the states of, representations of participants and Platform data.

### 4.3.3 Entities

An Instance can include two types of entities:

- *Users* are agents, such as humans or bots
- *Resources* are objects or actions that Users can affect

Elinor Ostrom stressed the importance of clarifying the bounds of the membership and the domain of a self-governing community. In Modular Politics, each entity has a state in the Instance and contributes to the state of the Instance as a whole. These entities can also become members of the Orgs in their Instance, which makes their states subject to the Modules that govern any Org they join. Entities may be assigned to particular Orgs by the Platform operator, or through voluntary agreements among Users. Users can assign that Resource to a member or members in that Org. If the Org is nested within another Org, entities must belong to or be added to the parent Org in order to join.

Each player on a Guilded Age server is a User; the server administrator decides the criteria for joining. The Guilded Age software correlates Resources in the Modular Politics Instance with the elements of gameplay, such as tools, skills, and

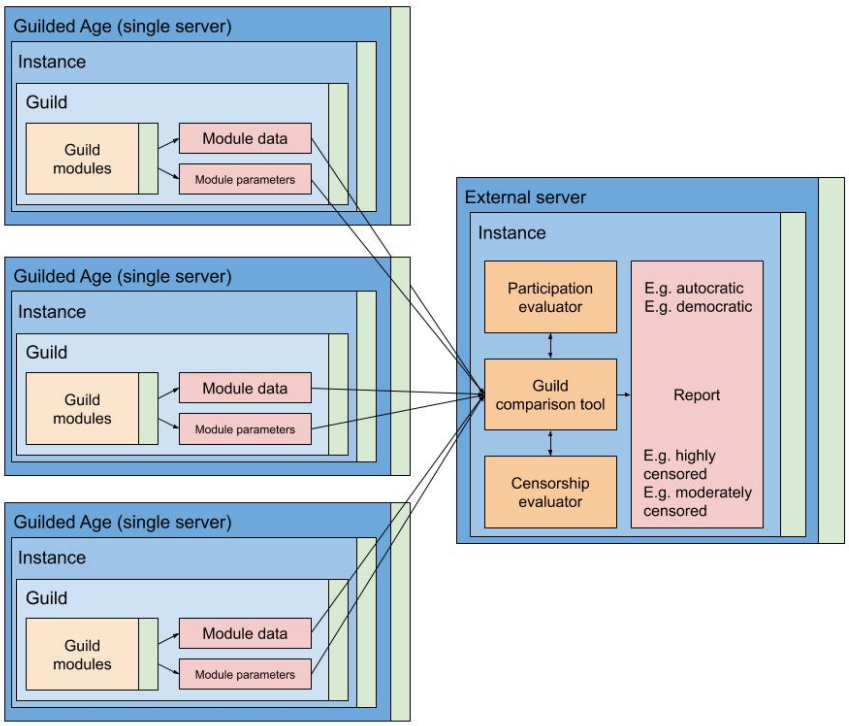



**Figure 4.3.2:** An example of a Monitor working across different Instances of Modular Politics. Continuing the Guilded Age example, here an external service queries a set of Modules shared between a set of guilds and then compares those guilds based on the data obtained from those Modules.

raw materials for monument-building. The administrator may choose to include a Module that allows players to create and layer their own Orgs.

Thus, the Instance configuration determines the relationship between Users and Resources in Modular Politics and the Platform. This relationship includes an identity management system for Users, as well as establishing any correlation between Resources and their manifestation on the Platform. To achieve the design goals of expressiveness and portability, it should be possible to connect virtually any kind of computational object a Platform might have into a User or Resource in Modular Politics.

### 4.3.4 Monitors

*Monitors* are "read-only" Modules that provide feedback by evaluating specific conditions according to consistent criteria. Their analyses can take place over a

certain duration or based on a snapshot at the time the Monitor is queried.

Monitors can gather data from throughout the system, such as by querying the states of specific Modules, Users, and Resources, or by querying the Policies that specify how a Module behaves. In Guilded Age, a Monitor might be able to query the tools and skills that players in one's guild possess and compare them to players in other guilds (Figure 4.3.2). Another Monitor might observe the rate at which each guild is building monuments. A guild could also install a Module that, when its monument-building falls behind a certain percentile compared to other guilds, abolishes the breaks from working that players otherwise take.

Monitor outputs can consist of multiple data types, such as a binary value (whether or not certain criteria are satisfied), a fractional value (such as a percentage score based on partial satisfaction of criteria), or an array (multiple data points on a given query). These outputs provide information on the states and behaviors of entities, Instances, Orgs, or Modules. They can also acquire considerable complexity by integrating diverse data sources or aggregating multiple other Modules through nesting. As a special type of Module, Monitors are intrinsically composable and thus contribute to the larger objective of modularity. A sophisticated Monitor could make broad claims about the nature of an Org, akin to indexes that evaluate the democratic health of governments [201, 202]. Because Monitors' source code and Policy configurations are available for inspection, their analytic methods should be in principle transparent.

### 4.3.5 Permissions

Access and authority in Modular Politics may not be equally distributed. Permissions are part of the configuration Policies for any Instance, Org, and Module. These permissions specify what actions Users can take in a particular environment. A User can create a new Org within an Instance, for example, only if the Instance's Policies grant that User the ability to do so. A Module can only query the state of a given Org if the User triggering that Module has obtained the necessary permissions to query that Org. Ultimately, all permissions in Modular Politics derive from those

that the platform administrators specify at the level of the Instance. A Guilded Age server might grant all users the right to use an Org-creating Module, for instance, or it might reserve that right for players who have been elected as masters of their guild.

In addition to specifying permissions within a given Instance or Org, a Module's Policies state whether it allows external API calls from other Instances, which may be on the same Platform or another. Such permissions are necessary in order to ensure that the goal of interoperability is handled responsibly. Moreover, a Module might be configured to allow for certain functions to be called only by specific Users, or it might allow access based on Users' membership in certain Orgs or Instances. Since Guilded Age is intrinsically a networked game, the software requires API access to basic functions, though individual guilds can decide whether it allows activities like interactions with players in different guilds.

Restrictions defined in a particular Instance or Org are automatically applicable to every Org created within it; permissions are inherited hierarchically. For example, if an Instance forbids external API calls, none of its Orgs can accept external API calls from Instances on other Platforms. Conversely, if an Instance is configured to accept external API calls, it will be possible for external Modular Politics Instances to interact with entities and Modules in the Instance—assuming that such interaction isn't elsewhere restricted. One Platform's Instance, consequently, could provide governance services to other Instances through APIs.

In Guilded Age, a guild might hold certain tools in an Org that only players with a certain rank in the guild can join. A Monitor, also, might be able to access certain information only if the player triggering it holds a given rank. And the Module that sets the players' rank may rely on the votes of only high-ranked players to carry out a promotion.

### 4.3.6 Further Configuration

Just as IAD emphasizes the need for participants to shape their governance environments, Modular Politics aims to provide wide latitude for configuring a governance environment without requiring extensive technical knowledge. Platform

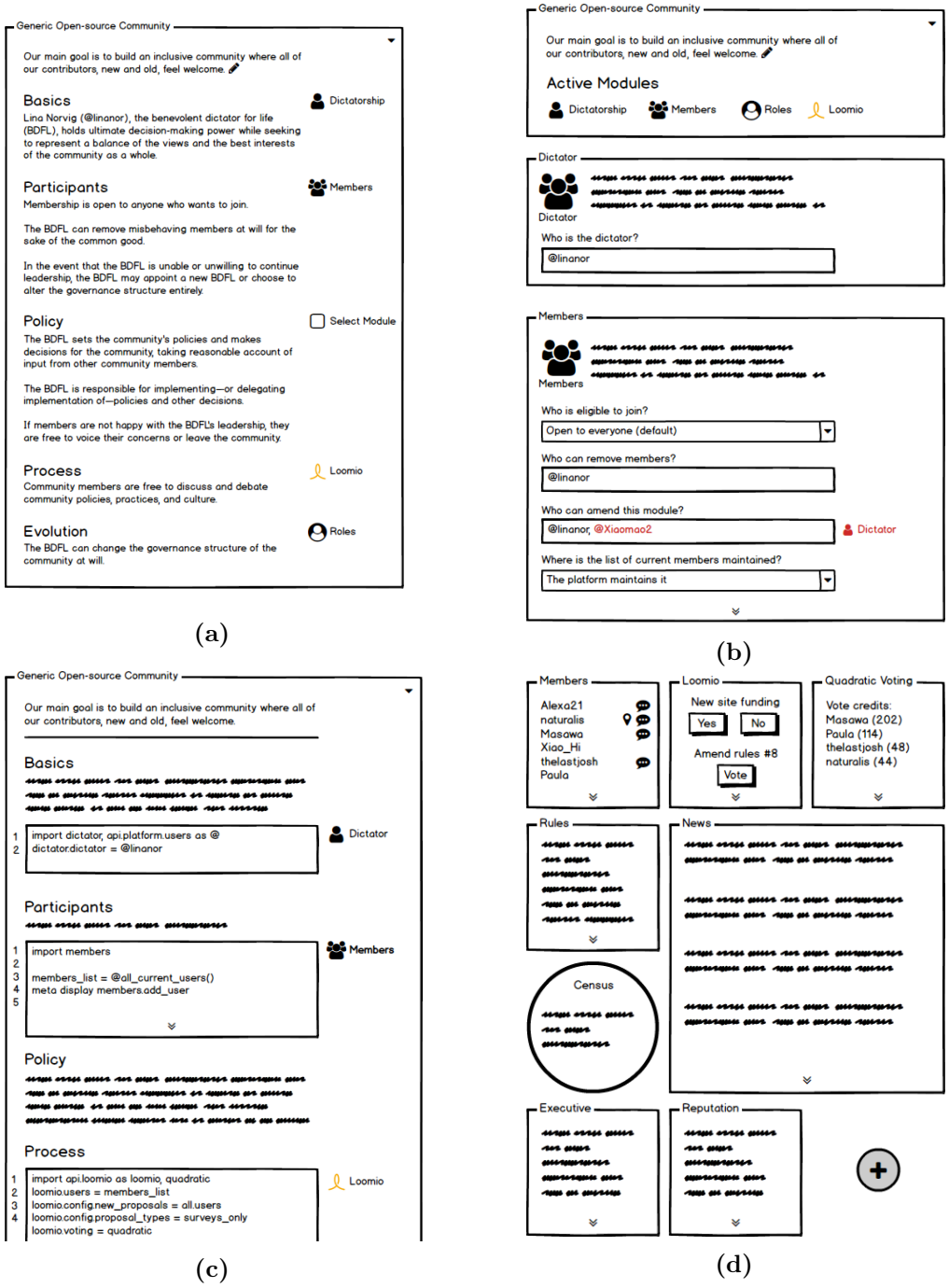
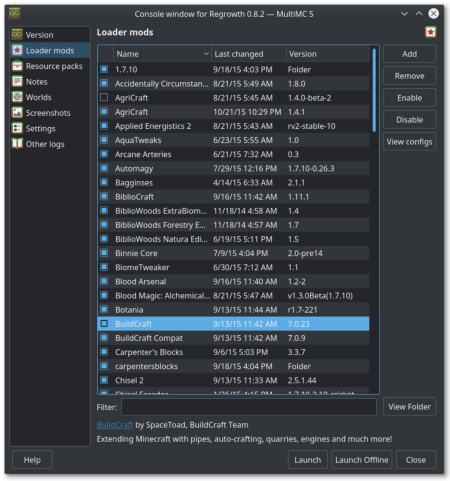



**Figure 4.3.3:** A set of example configuration interfaces for Orgs and Modules. (a) depicts a text-only representation that approximates an Org's textual constitution or code of conduct, (b) depicts only the actual configuration options of Modules along pre-defined parameters, (c) interpolates text with code snippets in a notebook-style interface, and (d) presents a possible end-user interface produced by the underlying Modules. Note that configuration Policies could also be made through visual programming or a simplified palette of options.

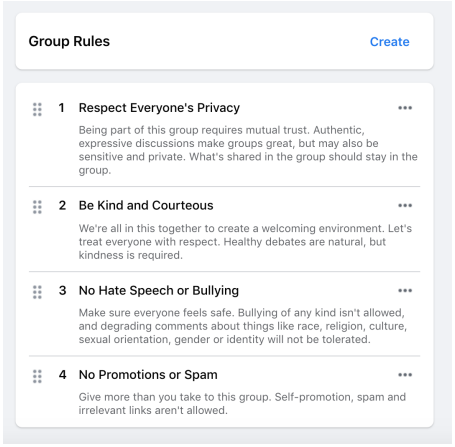
operators and their Users, if they have the necessary permissions, can create and customize their own governance systems. Configuration of these systems occurs at the level of the Instance, Org, and Module. Some Instances may have all their Orgs and Modules predefined by the Platform operators, with no opportunity for change or configuration by Users; other Instances may enable Users to modify the arrangement of Orgs and Modules as they choose. Operators can specify a fixed set of Modules available to their Users, or they can allow users the flexibility to acquire their own Modules. Discrete Platforms and communities using Modular Politics should be able to share, copy, and modify each other's Modules and Policies through public or private code repositories. Meeting the design goal of portability will be essential for accelerating governance innovation among users.

Configuration is essential in Guilded Age, as it is what enables guilds to define their own structures and rule-sets. Some guilds will allow configuration only by administrators, while others will see value in encouraging self-governance and creativity among players.

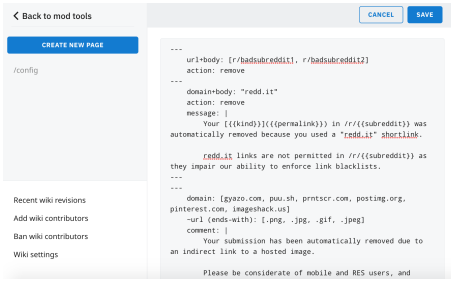
Moreover, to enable advanced configuration, Modular Politics would benefit from the use of software development kits (SDKs), which can be both internal or external to a given Platform (Figure 4.3.3). Guilded Age, for example, could implement an internal configuration tool available to server administrators and players. Through an SDK with an intuitive interface, Platform operators and participants alike might browse existing Modules or create their own; they might also adapt, customize, and debug these Modules before importing them into an Instance. SDKs could offer such features as a visual scripting editor [203] or a computational language [204] to support the development of fine-grained governance structures regardless of a person's technical skill level. Ensuring that Modules are portable, enabling users to adopt ones created for other platforms, will mean that people lacking the technical skills to create their own Modules can nevertheless choose and adopt tools created by others.



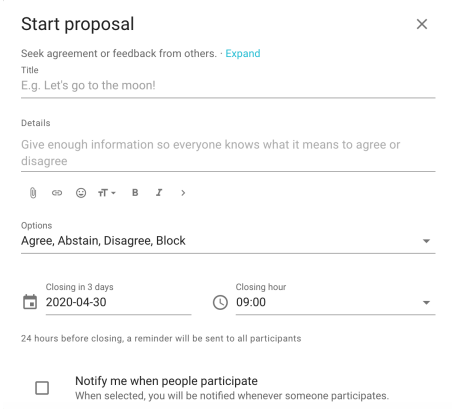

(a)



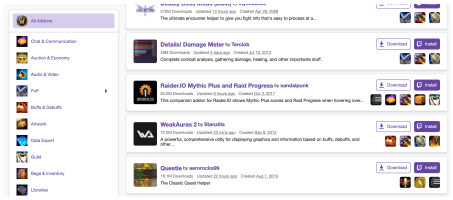

(b)



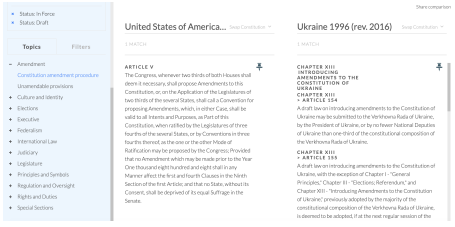

(c)




(d)



(e)




(f)

**Figure 4.3.4:** Examples of interfaces analogous to the kinds of configurations possible in Modular Politics. (a) MultiMC, a user-developed, open-source server manager and mod manager for Minecraft, (b) interface for setting rules for Facebook Groups; the rules here are templates suggested by Facebook, (c) Reddit Automoderator, a simple interface for implementing automated content moderation rules, (d) proposal settings on Loomio, a collective decision-making platform, (e) Curseforge, an addon manager and website for addons to several games, including World of Warcraft, (f) the Constitute Project, a collection of historical and in-force constitutions organized by an extensive ontology of terms.

### 4.3.7 Interface and Experience

Any Instance of Modular Politics must operate within a Platform, which can range in complexity from a simple Web server to a cloud-based multiplayer game or a blockchain protocol. Platform operators will have to implement strategies for defining and circumscribing the role of Modular Politics in the context of the host system, restricting the scope to particular participants and spheres of influence. Operators will also need to define how any particular Instance of Modular Politics interacts with their own systems. For Modular Politics to be effective, it should blend into its environment. It should also support and encourage best practices for accessibility and universal design wherever possible.

Modular Politics does not specify the details of interface or appearance, enabling developers to adapt the system to their needs. The developers of Guilded Age would have wide latitude in specifying how players experience the game, including its governance features (Figure 4.3.4). Yet, for the sake of clarity and consistency, Modular Politics should enforce some basic, shared logics of interaction. The design goal of portability should extend not just to Modules, that is, but also to user experience. Developers should expect to accept certain limits in order to benefit from the cross-platform Module ecosystem of Modular Politics. A person who gains proficiency using a Modular Politics system on one Platform will hence be able to transfer that proficiency to another Platform.

### 4.3.8 Implementation Strategies

We have outlined in general terms how Modular Politics might operate. But we have not specified the particular form it should take in software. Any implementation should be capable of meeting the goals stated at the outset: modularity, expressiveness, cross-platform portability, and cross-platform interoperability.

One approach to achieving these goals would be to offer Modular Politics solely through one central cloud-based service, which various Platforms would call upon over an API to handle their own governance processes. Such a centralized approach could provide a single, convenient repository of vetted Modules and assure the

consistency and reliability of all hosted Instances. Centralization could also help fund the development of Modular Politics, since access to the API might require paying fees to the organization that manages it. Yet some organizations might be reluctant to trust an external authority to provide a service as crucial as governance.

An alternative would be to define Modular Politics itself as a decentralized application running on a distributed-ledger technology (DLT); various Platforms could act as nodes on the protocol without delegating control over these processes to any central authority. But relying on DLTs might also introduce scalability and usability limitations; many such systems have not yet proven ready for widespread adoption. Furthermore, DLTs usually depend on token economics that could impose financial barriers on the adoption of Modular Politics. The economic imperatives of DLTs would also significantly constrain the expressiveness of Modular Politics for diverse types of institutions. While Modular Politics should be implementable on DLTs, it should not presume DLT infrastructure or that of any other specific type of software platform.

We believe the most flexible strategy would be to define an open standard, which can be implemented through a variety of open-source software libraries (perhaps resulting in a full-fledged software framework). The open standard would define the features and behaviors of any Modular Politics system, allowing for interoperability even when the actual implementations differ. Modular Politics could thus enable many kinds of Platforms to develop in tandem, through both replication and direct interaction [205]. The modularity of Modular Politics thereby occurs not just within a particular community but potentially through polycentric relationships among them.

By creating an open standard for a governance protocol, there would be no need to depend on a single organization's API or rely on the kinds of "cryptoeconomics" that DLT systems require—although both would be possible to implement. Guilded Age might depend on both a central server for coordinating gameplay alongside peer-to-peer interactions among servers. Additionally, because our design goals of cross-platform portability and interoperability are more complex and less essential

for some use-cases, the standard should allow for implementations that don't include all the features described here. As with open-source projects like Debian and Python, the development of the standard and its libraries should be community-governed, presumably using Modular Politics itself. Through such distributed collaboration, it is our hope that Modular Politics could become a widespread and extensible layer for governance on the Internet.

## 4.4 Use Cases

In order to further elucidate how Modular Politics might operate in practice, we illustrate its scope with two more specific scenarios in which such a system might be employed. The purpose of these hypothetical examples is to illustrate the range of institutions that a successful Modular Politics could serve.

The participants in these scenarios experience action arenas based on a set of pliable, co-determined governance conditions. Participants encounter an interface that enables them to navigate among the various features of the system, take action, and modify the system. Rather than adopting *ad hoc* governance practices within a rigid set of constraints, participants have the capacity to develop and employ tools appropriate to their communities' needs. Particular Modular Politics systems also have the option of interoperability with other systems, sharing governance mechanisms and corresponding data.

### 4.4.1 Example 1: Social-media moderation

Consider an affinity-based group on a social-media platform in which several thousand members share and discuss news about sculpture (Figure 4.4.1). The founders of the group adopted a simple system that enables members of the group to create governance proposals and pass them with a referendum-style vote by a majority of active members.

A controversy arises because some members begin posting pictures of sculptures that other members consider obscene and do not want to see in their feeds. Members begin proposing rules about what kinds of content should not be allowed, and several

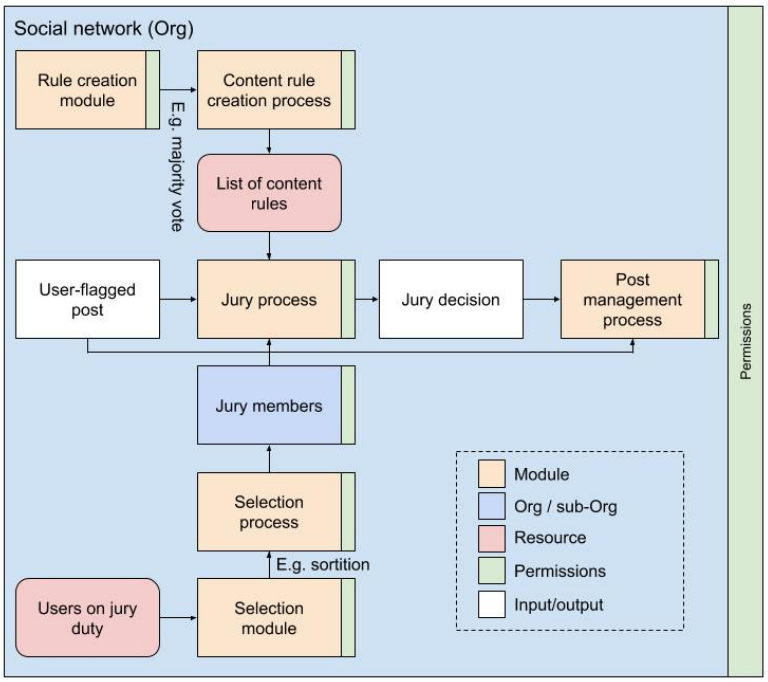



**Figure 4.4.1:** A schematic implementation of the system described in Example 1. Here, a community on social media implements a digital jury system for content moderation.

proposals pass by a majority vote. The founders, who have moderation authority by default, attempt to implement the rules by removing offending posts. But members begin to complain that the founders are being sloppy and non-transparent in how they interpret the rules. One member proposes to add a new tool that implements juries in the group, and the proposal passes. After that, whenever a user flags a post as offensive, it disappears and the platform queries five randomly selected members who previously volunteered for jury duty. If jury members support the removal of a post, they have to specify which rule or rules they believe it has broken. If no members object, the post is removed permanently. Otherwise, it reappears in the group.

This system satisfies most members, but a few raise concerns that the moderation has turned into overly zealous censorship. Although they cannot convince a majority to rescind the system, they do succeed in passing a proposal for a bot that automatically displays statistics on how many posts were flagged and how the juries voted on them. This feedback helps reveal which rules seem most

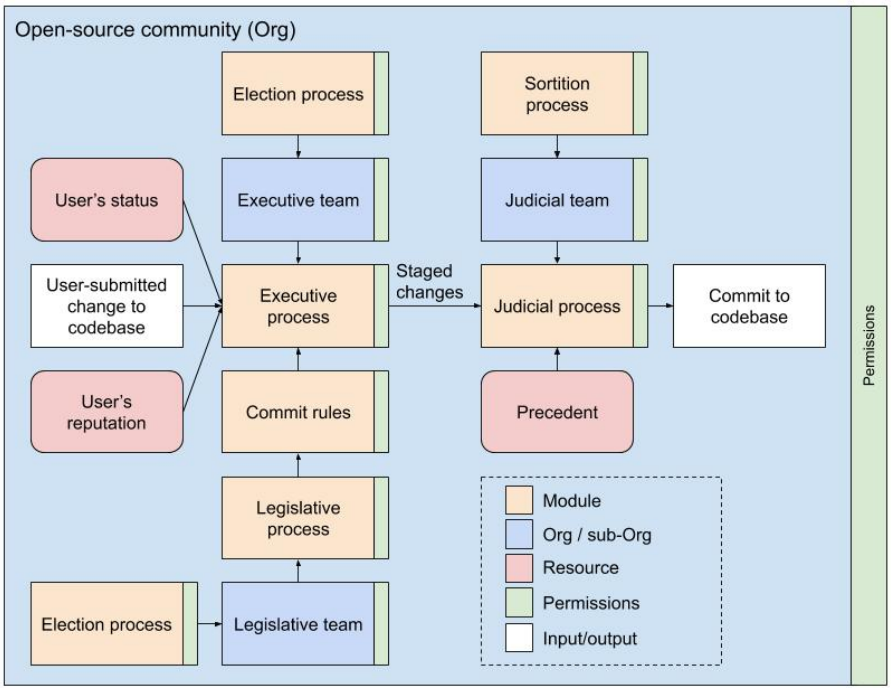



**Figure 4.4.2:** A schematic implementation of the system described in Example 2. Here, an open-source community implements a governance process for code commits involving several modules, ranging from those that specifically govern commits (commit rules) to those that govern who decides the commit rules (election process, sortition process, judicial process).

ambiguous to juries, spurring the development of more refined rule definitions and, eventually, lower rates of removal.

### 4.4.2 Example 2: Open-source software project

Consider an open-source software project that produces a popular, privacy-focused mobile operating system (Figure 4.4.2). Its founder is a developer who comes to be regarded as a "benevolent dictator for life," or BDFL. Although she is generally respected, her unvarnished leadership style frequently distracts the community from its core development work. The project hosts its codebase at BitGit, a decentralized, blockchain-based repository protocol that supports Modular Politics. In one of her moments of exasperation, the BDFL assembles a governance system that resembles a liberal democratic government, with distinct, elected bodies for establishing policies (legislative), implementing those policies (executive), and overseeing the

implementation (judicial). To vote or hold office, a developer needs to hold a staking token obtained from making at least one contribution incorporated into the codebase during the past year.

The former BDFL at first refrains from seeking office in the new system, but she doesn't exactly relinquish leadership. Her charismatic authority continues to hold sway, and the new elected leaders complain that they don't have sufficient mandate to carry out their roles and expect community support. They introduce a monitoring system into the project's BitGit interface that shows statistics on participation in the governance process, both for the project as a whole and for any given individual developer. The system also compares these statistics with those of other software projects also using Modular Politics on the BitGit ledger. This puts public pressure on developers to participate more actively in governance, and when they do, they tend to listen more to each other than to the BDFL's opinions. Eventually, the BDFL runs for a seat on the legislative body and is elected, thus channeling her role through the system she created.

## 4.5 Discussion

Modular Politics, as it has been presented in this chapter, is a strategy for enabling online communities to experiment with a wider variety of governance structures than is typically possible on existing Internet platforms. It draws on the logic of complex, polycentric action arenas as described in IAD research. Beyond simply replicating governance practices found elsewhere, online or offline, the model enables users to investigate novel opportunities for governance in the digital realm. Modular Politics could enact governance models that do not (and perhaps could not) exist in the offline world, inviting platform operators and their users to devise novel organizational practices for themselves.

On the one hand, we hope Modular Politics might enable online communities to adopt democratic practices more often than they presently do. But regardless of the systems they adopt, we suspect there will be benefits to making governance a more explicit feature of communities' experience, so as to prevent them from falling

into a "tyranny of structurelessness" [206]—a tyranny that has made no exception for technologists [207]. Our aim is not to make a moral or normative case for how online governance should be, but rather to empower all types of communities to develop experiments in governance and practice institutional artisanship.

For researchers, the computational nature of Modular Politics provides a means for studying the emergence and effects of various governance systems, including novel ones. User data has the potential to inform an ongoing conversation with existing social and political theories. The model already provides for reflecting back to users significant data on governance processes. Where appropriate, and according to policies set by platforms and communities themselves, this data could enable social scientists to generate new and more useful taxonomies of governance practices that, if properly analyzed, could contribute to improving the governance practices in various contexts. Beyond the immediate, practical benefits of such research, evidence from Modular Politics could contribute to the development of more robust theories of computational governance.

If implemented as an open standard, rather than as a centralized platform, Modular Politics could contribute to furthering the vision of what has been called "Web 3.0"—an open and interoperable network of online services that resist the tendencies of centralization that have become so persistent on the Internet today. In the context of governance, this means that a particular community could rely on a variety of integrated services in order to achieve a particular governance structure that spans across multiple platforms. For instance, a community developing open-source software could choose to combine the reputation system of the social news platform Slashdot and moderation standards of a particular group on Reddit in order to assign influence to different users within the community's governance structure. Deliberation could occur on a Discord chat, whereas decisions could be made on the Loomio platform. Once a decision has been made, the implications could be automatically enforced across the relevant platforms. An open standard for governance could thereby contribute to the emergence of specialised services designed to respond to the needs of different communities, while maintaining

a high level of integration across these services in order to ensure a cohesive experience of community governance. It is our hope that such a model will inspire users to develop ever more sophisticated governance techniques, which others can adopt in their own contexts.

In addition to fostering new types of online governance innovation and experimentation, we hope to see the effects of Modular Politics extend beyond online communities to political cultures more broadly. If Modular Politics is successful in accelerating the pace of governance innovation through experimentation, the resulting insights can be instantiated in a variety of communities, regardless of the medium of choice. Whereas people typically experience governance in workplaces and governments as fixed and remote, Modular Politics could make the practice of designing and testing governance systems far more widespread, and contribute to the revitalization of civic engagement that the Internet has so long promised and struggled to deliver. While many people's experiences of political participation tend to be limited to such mechanisms as electing representatives or voting in referendums, Modular Politics could expose them to other forms, such as delegative voting systems or sortition. There is surely also a wide spectrum of alternative governance models that have yet to be conceived. Online communities could become powerful laboratories testing new ideas and learning from the failures and successes of other communities, potentially spurring a renaissance in democratic culture.

## 4.6 Limitations

Modular Politics, along with our treatment of it here, has a number of limitations. We begin with its limitations, indicating future work.

First, we have argued for the importance of governance innovation throughout this article but have not proven (conceptually or empirically) that it would be easier to innovate in Modular Politics than otherwise. We believe that modularity and portability, in particular, will accelerate innovation—e.g. because they lower the costs of innovation, reduce the risks of failure, create more opportunity for creativity, and help foster ecosystems that will consume the products of new innovation—but

there are many factors (e.g., software usability, software variability or malleability, programming language design, simulation and inference tools) that could help or hinder the effectiveness of Modular Politics as a tool for innovation. We flesh out the larger problem of governance innovation in Chapter 7.

Second, we have not defined exactly what kinds of new governance systems and structures would and would not be expressible in Modular Politics. For example, future research that takes a more mathematical approach might define an explicit space of governance structures and prove that Modular Politics is able to recover all elements in that space. This is partially developed in Chapter 2 and more explicitly in Chapter 3 as part of the analysis of the space of contracts. However, a rigorous completeness result would be particularly exciting and could motivate significant theoretical progress. To obtain such a result, we would need to (1) define a sufficiently broad class of governance structures (as a subset of all institutions), (2) define what it means to have a "modular" model of such governance structures, and (3) show that the model can generate all instances of the class under a set of well-defined operators. We preview some future work in Chapter 7.

Third, although Modular Politics is intended to support heterogeneous institutional forms, our formulation surely bears built-in political biases that we have yet to interrogate. All technological artifacts such as this carry political qualities, whether because they tend to be used for certain political purposes or because their design inclines toward a certain political system or philosophy [208]. One-size-fits-all technical solutions run the risk of imposing a false universalism [209]. We hope to learn from early user testing how to ensure that Modular Politics facilitates bottom-up governance innovation rather than imposing norms. We also have yet to explore how commercialization and economic incentives might affect the use of Modular Politics in practice.

The above points address the limitations of this paper and its methodology. Modular Politics, as a model or as a technical specification, has a separate set of limitations, indicating fundamental constraints and issues of scope.

First, Modular Politics is not intended as a proposal for offline community governance. While we believe that online experiments will eventually teach us much about about offline governance, we have chosen to narrow the scope of our present inquiry to communities that exist and interact mostly online through digital platforms and messaging services.

Second, Modular Politics does not constitute or aspire to be a complete theory of governance. It is, rather, a proposal for tools that might support various forms of governance within a digital "state of nature," however fictional such a state inevitably is. The model can, however, serve as the basis of theoretical claims and empirical tests about the system and the behavior of its users. We also understand our proposal as a complement to various efforts that introduce more accountable governance into the ownership structures, business models, and public policies for online platforms [210].

Finally, and of particular importance, we emphasize once again that constructing an effective system of governance requires more than just software. Cultural and contextual factors are also essential [211]. Modular Politics does not specify, but depends upon, a broader "civic sphere," by which we mean the confluence of tools, culture, and context through which participants interact. A healthy civic sphere is one in which participants feel confidence that the governance process is meaningfully accountable to them [212], with some kind of shared norms, appropriate information flows, and the power to influence decisions. Crafting the civic sphere is the job of platform operators, community leaders, and community members as a whole. For example, a particular community might adopt language that makes governance activities appear fun and silly, or alternatively grave and serious. Policies might require prospective participants to agree to act in good faith, or other prerequisites, before joining the community. Platform operators might choose to implement Modular Politics in ways that bear biases toward more or less democratic practices. Users in positions of authority might choose to emphasize negative sanctions for rule-breakers or positive reinforcement. These kinds of choices are vitally important, and

Modular Politics does not prescribe them. Modular Politics, as with any procedural or computational system, is no replacement for other aspects of a healthy civic sphere.

## 4.7 Conclusion

The tools available for governance in online communities are currently limited, inflexible, and ill-equipped for governance innovation. Modular Politics seeks to encourage such innovation through a dynamic, flexible model for online governance, through which community members can engage in creating and experimenting with a variety of different governance techniques. As such, Modular Politics could accelerate and proliferate innovation in governance design well beyond what occurs in offline systems, which carry considerable burdens of inertia and path dependence. Ultimately, a successful open standard for governance could contribute to making creative and responsive governance a more widespread norm—both online and offline.

This chapter is a preliminary step toward that goal. Much research and experimentation remains in order to define a standard that is both durable and attractive for platforms to adopt, which would require specifying Modular Politics in much greater detail than we have done here. We hope at least to have spurred interest in the challenges to come.

# 5
# Agreement engine

How are digital agreement systems designed and implemented, and how can we help make it easier to design, deploy, and extend such agreement systems? Chapter 4 proposed a conceptual framework for governing online communities, but did little to theorize or implement that framework. In this chapter, we build a conceptual model of digital agreement systems as well as a practical software library for deploying such agreement systems, and test the model and library on two live experiments.

To add extra emphasis: the subject of this chapter is *systems*. It thus offers an orthogonal, complementary perspective on contracts to the game-theoretic perspective developed in Chapter 3—instead of engineering the contracts themselves, in this chapter our goal is to engineer the systems that deploy and interpret contracts.

## Contents

## 5.1 Introduction

Agreements are the building blocks of modern societies. Whenever you want to work with other people—whether it's starting a business, building open-source software, or slaying a (virtual) dragon—you will need to first form an agreement with those people to communicate expectations, safeguard rights, or divvy up rewards. When we change the agreements that people can form, for example through policies that affect communication technology, social trust, or the rule of law, we change the kinds of economies and politics that people produce.

But what determines the agreements that people can form? Or, to put it another way: for a given kind of agreement (legal, informal, computational, etc.), how can we build *systems* for authoring and enforcing such agreements? Such agreement systems range widely: from a state's business registration process, to the small claims court system, to the Ethereum smart contract system, to the guild systems of online games like World of Warcraft or EVE Online, to informal systems of agreement that govern everything from bazaars to traffic circles.

In this article, we introduce agreement paths, a general formalism for modeling agreement systems, and Agreement Engine, a software service for building net-native agreement systems. In doing so, we emphasize (1) the growing importance of agreement systems that are implemented partially or wholly online and (2) the benefits of a more rigorous and interoperable framework for engineering such systems. We also present two experimental agreement systems, Scarce Knowledge and Twitter Social Capital, built using Agreement Engine. Finally, we contextualize our contributions within the larger project of building net-native agreement systems and close out with a discussion of the user experience of legal contracts compared to smart contracts or informal agreements.

## 5.2 Background and related work

What do the following have in common?

1. A signed rental contract
2. A one-line email: “cool, meet you at 9pm outside the bar"
3. A (multi-party) smart contract
4. A bid on eBay
5. A video of two people exchanging nods

Answer: they are all manifestations of mutual assent, a.k.a. *agreements*. *Mutual assent* is just a piece of jargon that means “we agree"—it’s an abstract relation (what the law calls a ‘meeting of the minds’) between two or more subjects that share a common intention or belief. Agreements are *manifestations* of mutual assent, meaning that they bear witness to the claim that a common intention or belief exists [213]. To put it another way: every agreement constitutes evidence of mutual assent, though not every such agreement is thereby made enforceable.

Many agreements also incur *obligations*, where an obligation is an action that one has to do. These obligations usually assume certain mechanisms for *enforcement*, where *enforcement* is the act of compelling compliance with an obligation. For example: an email thread between two friends agreeing to meet at a bar (1) is explicit evidence of mutual assent and (2) creates a social obligation to meet at said bar where (3) the obligation to show up is enforced by an implied social norm: don’t break promises to friends or you’ll lose your friends and/or be reputed as a flake. A bid on eBay, on the other hand, (1) is evidence of the bidder’s agreement to buy the item at that price and (2) creates an obligation to pay the seller at that price if they win the auction where (3) that obligation is presumptively legally enforceable (4) either through eBay’s automated escrow system or through eBay’s (less-automated) unpaid item policy. [214]

This (1)-(2)-(3) pattern of agreement-obligation-enforcement is extremely common in society and in the law. We call this pattern a *contract system*, and we call agreements that fit this pattern *contracts*. There is substantial theoretical and empirical work on contract systems within comparative law [215], contract law [216], philosophy [217], and economics [6], mostly focusing on national legal contract systems. Social scientists have also studied informal agreement systems—informal only in the sense that they do not directly involve a court system—through an institutional lens. [12] And most recently, smart contract systems—systems of agreements automatically enforced through code—have matured from early conceptual models [63] into large, blockchain-based platforms such as Ethereum [218] and Cosmos [219].

In what follows, we will focus on a number of features shared between traditional legal contract systems, informal agreement systems, and smart contract systems.

| **Term** | **Definition** |
|---|---|
| Agreement | A manifestation of mutual assent between a number of parties |
| Obligation | An action that one has to do |
| Enforcement | The act of compelling compliance with an obligation |
| Contract | An agreement that creates an obligation with an enforcement mechanism |
| Contract system | A system for creating and enforcing contracts |
| Legal contract | An agreement recognized and enforced by the state, typically through a system of laws, courts, and enforcement agencies |
| Smart contract | An agreement written in code and enforced by a technical system such as a virtual world or blockchain |
| Informal agreement | An agreement not intended to be enforced through a legal or technical mechanism[1] but instead through social norms, reputational incentives, and/or extralegal means that do not depend on a formal institutional enforcement mechanism |
| Joint account | A digital representation of an n-person agreement, akin to a user account |
| Registration | The process of recording or representing a completed agreement and related data, usually by a qualified authority |
| Authentication | The process of checking the data, status, and/or registration of an agreement, usually by an enforcement mechanism |

**Table 5.2.1:** A glossary of the many terms in this paper.

[1]Informal agreements can include very explicit terms and consequences, and some are actually more diligently enforced than any formal contract, e.g. a loan shark's "if you don't pay me back, I'm going to beat you up". The difference is that there is no formal institutional structure or system for enforcement.

## 5.3 Modeling agreement systems

Imagine that you are driving through the streets of Delhi. A worn-down yellow tuk-tuk beeps; you slow, and it merges in front of you. At the next stop light, you wince as a big garbage truck rumbles to a stop just behind your back window. A young chai wallah weaves through the cars with a tray of plastic cups; you wave him over and trade him a few rupees for a cup. The tuk-tuk ahead of you moves on; you accelerate too, swerving past a bicycle vendor on the corner, and turn onto a crowded traffic circle. As you inch forward, a red hatchback edges up at the next turn, signaling right. You slow, graciously; the hatchback moves gingerly up until it works up the courage and enters ahead of you into the wide stream of cars heading downtown.

Every day, we negotiate agreements with each other. We signal; we gesture; we infer; we agree; we act. It is second nature to us. Most of these rich interactions do not produce well-defined artifacts in the form of enforceable contracts. But most *do* take place within systems that support and facilitate them, systems that have a passing resemblance, if you stare at them closely, to more formal systems like the courts. Our goal in this section is to understand and characterize some of these similarities.

We present *agreement paths*, a formalism for modeling systems of agreements that abstracts over the content and type of an agreement. While agreement paths are general enough to apply to many types of agreements, they are not intended to be a general analytic instrument for scholars of agreements in philosophy, economics, or law. *Agreement paths are intended to highlight the elements of an agreement system that can be digitized.* Our goal is not to present a general model for analyzing or classifying agreement systems but to model the components that are present in digital agreement systems like DocuSign Agreement Cloud, Ethereum, Aragon Court, or Kleros. This way we can more easily (1) recognize these digital agreement systems when they occur on the internet, (2) amend these systems as we amend institutions for agreement in the real world, (3) identify the places where digital processes can enter into offline agreement systems, and (4) build more modular and “swappable" agreement systems.

Every agreement path can be broken down into six stages: *authoring*, *registration*, *execution*, *authentication*, *appeal*, and *enforcement*. An authoring process allows parties to negotiate and author agreements with each other. A registration process records the data pertaining to an completed agreement, often including the agreement's identity, content, parties, and status along with secondary authentication data (e.g. the name of a witness). Once registered, the agreement is executed or enacted by the parties to the agreement. During execution, a party may trigger an appeal to exit the execution process for whatever reason.[2] Following an appeal, an authentication process verifies the data, validity, and/or registration status of an agreement. Finally, an enforcement process interprets the content of an agreement, often but not always with accompanying evidence, in order to enforce its execution or compensate for any outstanding obligations. See Fig. 5.3.1.

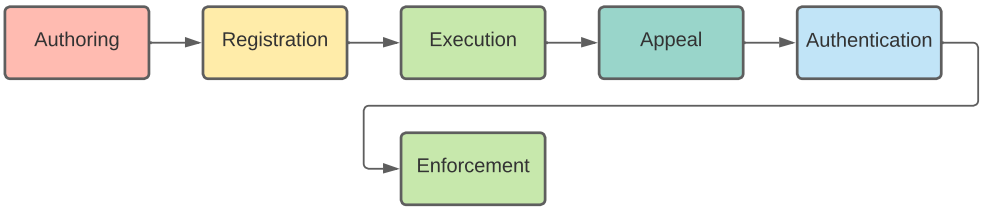



**Figure 5.3.1:** The six stages of an agreement path: authoring, registration, execution/enforcement, appeal, and authentication. This model will serve as a legend for many of the figures below: processes are colored red for authoring, yellow for registration, green for execution / enforcement, teal for appeal, and blue for authentication.

For example, Alice and Bob might negotiate the terms of an employment contract over email, trading versions back and forth (authoring). Once they are both satisfied, they convert the final agreement to a PDF and each sign it digitally, trading the signed and counter-signed copies over email (registration). With the contract signed,

[2]To be clear, most agreements self-execute without needing to trigger any external appeal process. Indeed, more complex agreements often define terms for mediation or resolution that do not involve any entities beyond the parties to the agreement. For example, a customer claims that a product is defective and the manufacturer sends them a new product without invoking a court or arguing about the warranty. An (external) appeal should be distinguished from actions defined within the content of the agreement for resolving disputes. More generally, an appeal (or the right to appeal) is a part of the meaning of the agreement that is independent of the "content" of the agreement. In other words, the agreement itself may not note any right to appeal or 'exit'. Appeals are enabled by the agreement system, not by any individual agreement.

Bob begins working for Alice, and in return Alice pays Bob a salary, as per the contract (execution). At some point, Alice stops paying Bob and stops responding to calls or email; Bob is forced to file a lawsuit to get back his wages (appeal). As part of this lawsuit, he appends copies of the emails and the signed employment contracts (authentication). The lawsuit enters the legal system. A small claims court rules against Alice, who does not show up to contest the lawsuit, and orders Alice to pay Bob (enforcement) by sending a letter to Alice's last-known address. Alice does not respond to the letter, at which point the trial court issues a contempt order for Alice to be arrested (appeal).

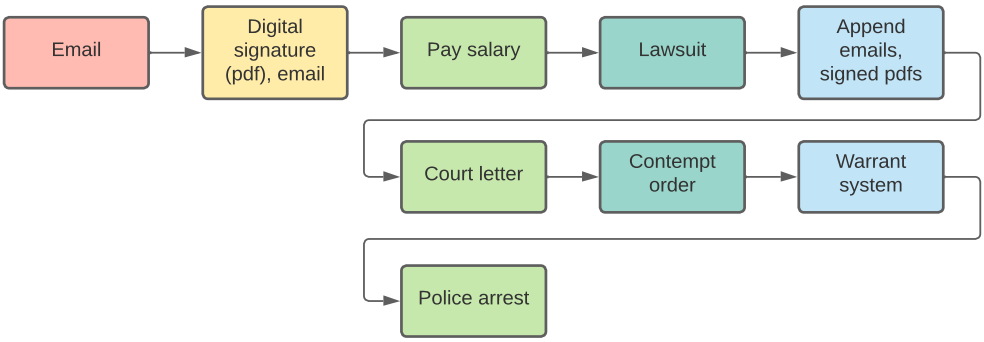



**Figure 5.3.2:** A simple employment contract negotiated over email and enforced via a trial court.

Not all agreement paths are as straightforward as the one above.

For example, Ghost Knowledge [220] is a website that allows people to crowdfund "bounties" for specific authors or researchers to write essays. Once pledges go past a certain threshold, Ghost Knowledge messages the authors on social media, who then either accept or reject the bounty; if they accept, the money is put into an escrow account to be released once the essay is published.

Ghost Knowledge was built and released using no-code tools including Airtable, PayPal, and the Twitter API. A substantial portion of the author interactions involved manual messaging by the founders. [221] We will come back to Ghost Knowledge in the next section, when we recreate a version of it using the Agreement Engine tool.

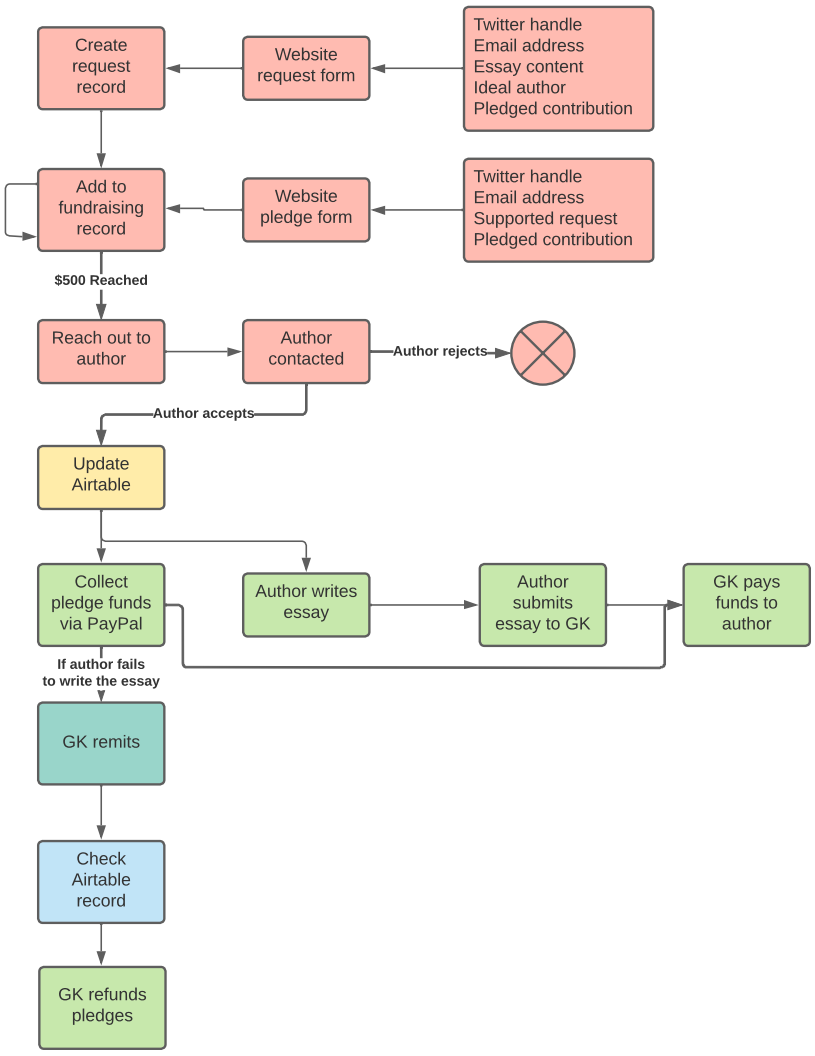



**Figure 5.3.3:** The agreement path of Ghost Knowledge, a website for crowdfunding written essays.

## 5.4 Constructing and composing agreement paths

Agreement paths, on their own, are a conceptual model intended to support the composition and substitution of different processes within agreement systems. In this section, we present Agreement Engine, a software tool for building net-native agreement systems. Agreement Engine implements the agreement path model.

To be clear, Agreement Engine itself is neither a system for authoring individual agreements nor a system for enforcing them. However, it has built-in interfaces with existing authoring and enforcement systems.

| Scope | Definition | Examples |
|---|---|---|
| Authoring | An authoring process allows end-users to negotiate and author agreements with each other. | Ghost knowledge negotiation (No-code form + Twitter DMs) |
| Registration | The process of recording or representing a completed agreement and related data, usually by a qualified authority | Signing a physical document, clicking accept on a digital form |
| Execution | A special layer of the enforcement process related to the content of an agreement. Includes things such as enactment, delivery, and any number of triggers specified in the content of the agreement, e.g. revisions, termination clauses, and so forth. | Payment, sending of goods, execution of contract calls |
| Appeal | A process that triggers an authentication and enforcement process. Appeals can be pre-specified within an agreement, but they always call an external process that is not specified in the agreement itself. | Communication between parties to revise a contract. *Non-example: an adjudication clause specified in the contract itself.* |
| Authentication | The process of checking the data, status, and/or registration of an agreement, usually by an enforcement mechanism | Admission of evidence in a courtroom, reporting a video on YouTube |
| Enforcement | An enforcement mechanism interprets the content of an authenticated agreement, often (but not always) with accompanying evidence, in order to enforce or compensate for any outstanding obligations. | Kleros jury, U.S. justice system, moderator intervention on a social media platform |

**Table 5.3.1:** Definitions of the stages of an agreement path.

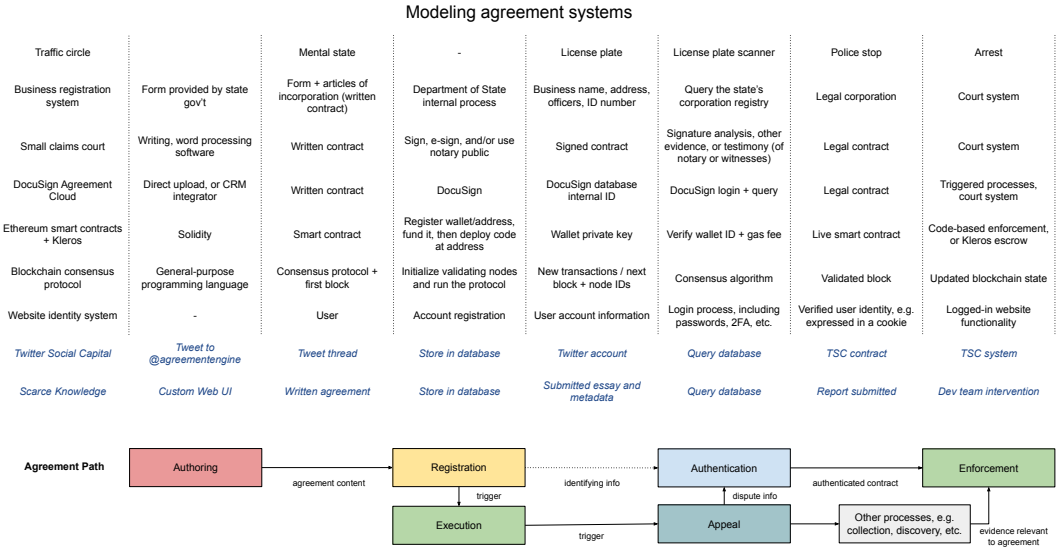



**Figure 5.3.4:** Additional examples of the key processes in an agreement system: *authoring*, *registration*, *execution*, *appeal, authentication*, and *enforcement*.

### 5.4.1 Agreement processes

From a top-down perspective, each agreement system created in Agreement Engine can be considered a state machine composed of many processes. Each process represents a distinct and independent service: authoring, registration, authentication, and so on. Within a given path, not all processes need to be computational, as long as they have a formal representation. When a process is active, all data input to the agreement is routed to it for processing. The process can then decide to wait for additional input, transition to another process, or terminate the agreement. Processes can also take action immediately when they are activated or right before the process ends, instead of asynchronously upon receiving input. Using this method, we can chain discrete processes into agreement paths which exhibit more complex behavior.

In our archetypical agreement system model, this starts with an *Authoring* process and ends with an *Enforcement* process.

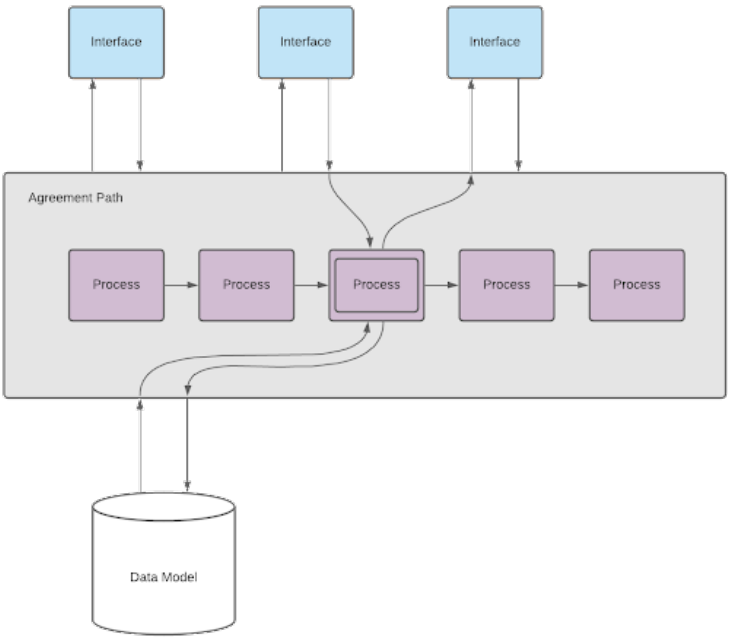



**Figure 5.4.1:** Diagram of a single agreement instance and connected systems.

| Feature | Notary public | DocuSign Agreement Cloud | Kleros or Aragon Agreements |
|---|---|---|---|
| Supports authoring | No | Yes, through Gen and other integrators | No |
| For legal consumption | Yes | Yes | No |
| For consumption by online community | No | No | Yes |
| Many enforcement mechanisms | No, just the law | No, just the law | No, just escrow |
| Many authoring platforms | No | Some, e.g. Salesforce | No, just Ethereum |
| Includes decision-making process for enforcement | No | No | Yes, digital jury |

**Table 5.4.1:** Comparison of different registration services.

### 5.4.2 Agreement interfaces and instances

Within Agreement Engine, agreement paths are high-level data structures that model a single agreement system along with an arbitrary number of agreements authored within that system. Paths can manage processes by keeping track of which process is currently active, handling state transitions, and providing access to each agreement's private data model. Agreement paths also filter and redirect agreement input and output via a set of agreement interfaces. Interfaces filter incoming data from various sources so that it can be preprocessed and routed to the correct agreement instance (or used to create a new agreement instance). As seen in Fig. 5.4.1, a single instance of an agreement path, or simply referred to as an agreement instance, acts as a passthrough for the currently-active process. It allows the agreement to be viewed as a single object from the outside, while facilitating more complex decision-making from the inside. Each agreement instance has its own state and data model in order to track the path it is associated with, the stage within that path, and any additional data relevant to its execution. Paths can manage any number of agreements, as seen in Fig. 5.4.2.

### 5.4.3 Agreement servers

Agreement servers allow for multiple paths (and thus multiple agreement instances) to be run at the same time. The server receives requests and routes them to the

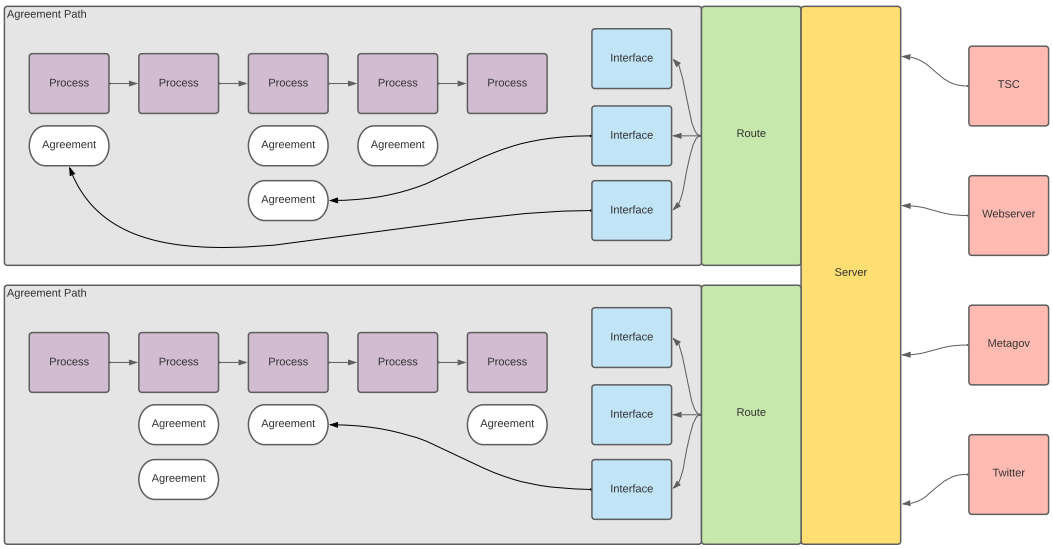



**Figure 5.4.2:** Diagram of a server with multiple paths and active agreements.

corresponding path where the interfaces decide which agreement instance should receive the data. This system allows for arbitrary data to be handled, whether it's from a local process, a webhook, or another web server. The server provides the infrastructure for agreements to connect to the internet and interface with other platforms and software. For example, suppose we have a server running on a web server with the domain name "*myagreements.com.*" We could configure a new agreement path called "*Employment*" for managing employment contracts. Now our agreements can receive data sent to "*myagreements.com/Employment*" via REST API calls from other applications, or webhooks from platforms such as Twitter or Slack.

### 5.4.4 Example: Scarce Knowledge

Let's look at an example agreement system built using Agreement Engine. This demo, which we call Scarce Knowledge, recreates a version of the Ghost Knowledge platform that we described in the previous section. Fig. 5.4.3 shows the agreement path of this version. Requests to Scarce Knowledge are initiated by a user-facing website with a simple form that forwards data to Agreement Engine. Once pledged contributions surpass $500, a process transition is triggered. This is done via a

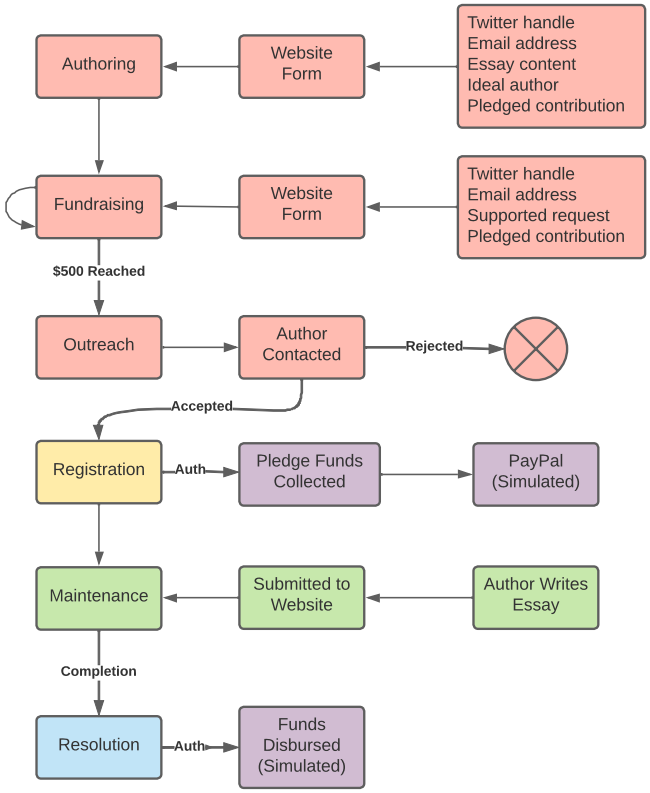



**Figure 5.4.3:** The agreement path for Scarce Knowledge. To help distinguish between execution and enforcement mechanisms in this particular case, we have highlighted the enforcement mechanisms in purple here. While these enforcement mechanisms occur outside of any particular Scarce Knowledge agreement, they are part of the larger Scarce Knowledge agreement system and are crucial for correct function. More generally, many digital agreement systems will incorporate calls or appeals to many, many external enforcement / execution services, blurring the lines between execution and enforcement.

Twitter interface that sends a direct message to the requested author. The next few processes return to the website to handle final essay submission. Finally, an email automation service is called to forward the essay to those who supported it. By using Agreement Engine, we were able to manage a complex decision-making process with several approval steps that span a user facing frontend, Twitter, and a mail automation service. In a production implementation this would also include PayPal integration. Fig. 5.4.4 below shows part of the user interface flow for this agreement path implementation using a simple website and Twitter direct messages.

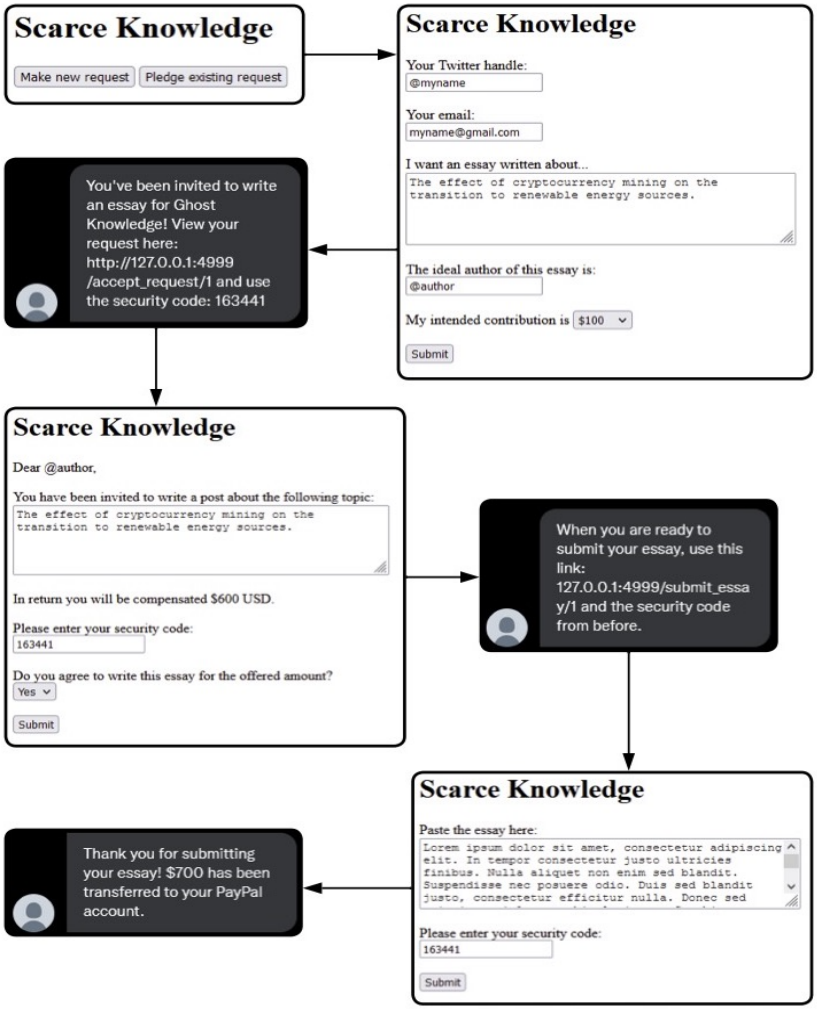



**Figure 5.4.4:** User interface diagram for “Scarce Knowledge”.

### 5.4.5 Example: Twitter Social Capital

Let’s look at another example agreement system implemented in Agreement Engine. Twitter Social Capital (TSC) is a Twitter-based agreement system that allows users to register and enforce agreements with each other by tweeting at the @agreementengine bot. New agreements are authored by tagging another user and using the keyword “agreement" Users can then reply to the tweet with “upheld" or “broken" to indicate the status of the agreement. If users disagree on the outcome they are given the chance to change their responses and come to a consensus. Finally, the bot publicly records the result of the agreement. As seen in Fig. 5.4.5, by using Agreement Engine we were able to create a branching and looping control flow to

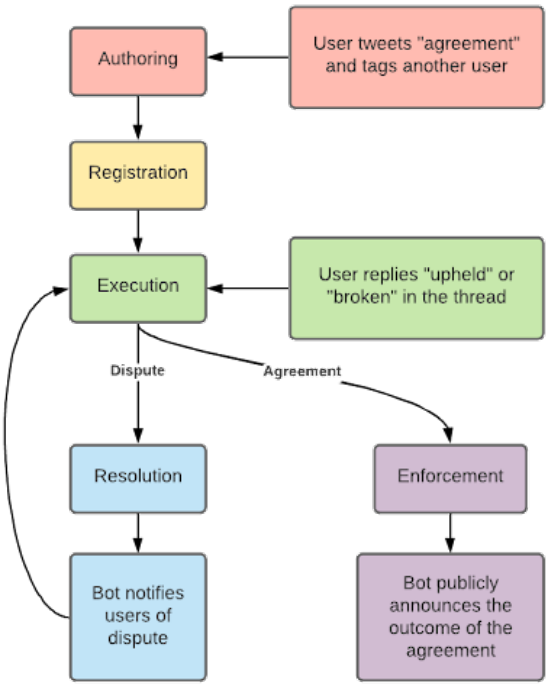



**Figure 5.4.5:** The agreement path for Twitter Social Capital (TSC).

resolve disputes and only terminate when an agreement is reached. This example uses an interface to the Metagov Gateway to send and receive tweets. The technical details of interfaces are discussed later, but this implementation demonstrates the flexibility of Agreement Engine in integrating with other platforms.

## 5.5 Using the Agreement Engine library

When creating an agreement system using the Agreement Engine Python library, there are three main components that should be considered: the server, paths, and interfaces. Let's construct a simple agreement to explore how this works.

```
class MyInterface(Interface):
    def filter(self):
        return True
```

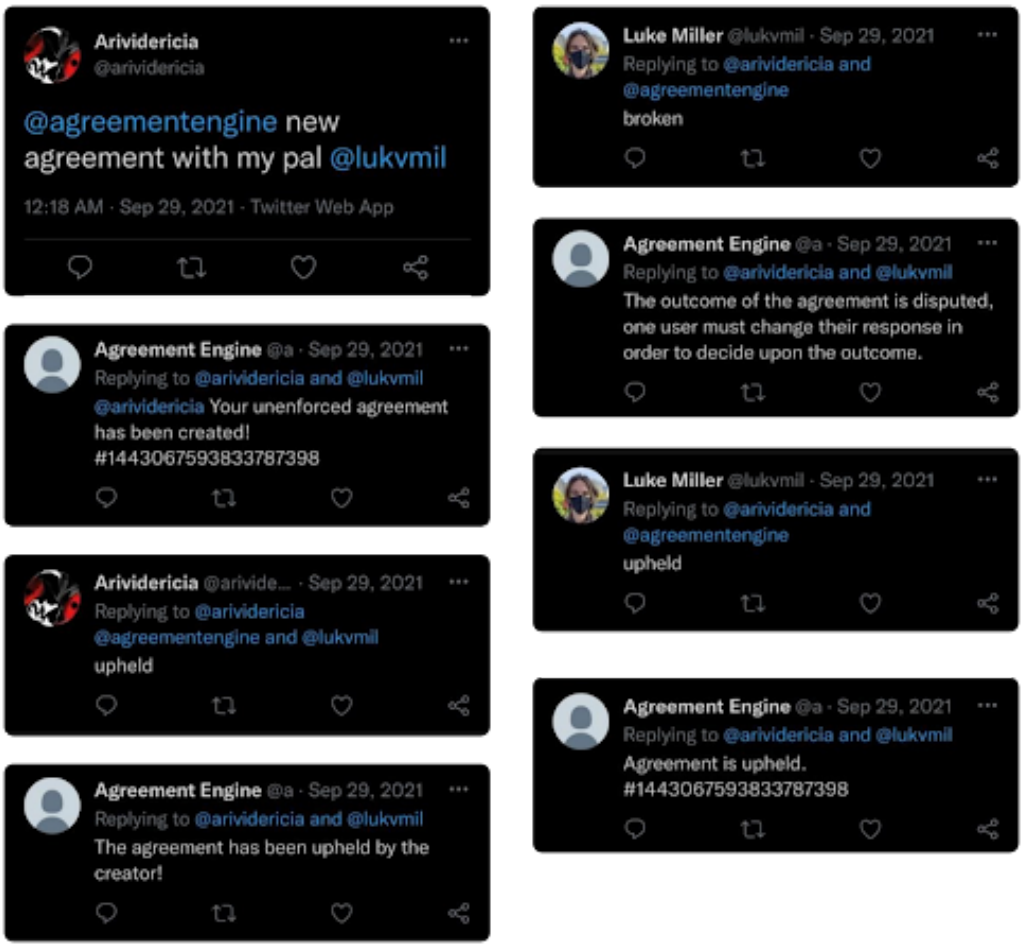



**Figure 5.4.6:** Example tweets from the TSC Twitter bot that we built using the Agreement Engine.

```
    def match(self):
        return self.new_agreement()
```

First we need to define an interface. This is how a path decides whether incoming data should be routed to an existing agreement or a new one should be created. In the example code above, we create an interface that handles all input, and always creates a new agreement. Real applications will decide when different interfaces should govern input data, and how to match existing agreements. For example, the Scarce Knowledge example uses two interfaces to communicate with the user facing web server and the Twitter API.

```
class MyPath(AgreementPath):
    class Authoring(AgreementProcess):
        def on_receive(self, data):
            self.model.set('data', 'received', data)
            self.path.terminate()

    interfaces = [
        MyInterface
    ]

    init = Authoring
```

Next we define our agreement path, this is the primary structure that represents our agreement system. It will contain all of our processes and define the path that an agreement takes through them. In this example we have a single authoring process that records the data we received to the agreement's data model (a simple JSON format with different fields) and then terminates. We also set up the interfaces that the path is using, and the initial process that a new agreement will start in. Paths can implement more complex logical flow such as branches or loops. Our example paths use four processes that can terminate early upon detecting invalid data inputs.

```
server = Server('db.json')
server.add_path(MyPath)
server.run()
```

Finally we create the server that actually receives and sends requests. All we need to do is define a JSON file to store our agreement data and add our paths. Servers can support any number of paths, as long as they have unique names. Each path will receive a unique API URL for receiving requests. The URL for a path on a local server will look like http://*127.0.0.1/MyPath*.

## 5.6 The challenge of composition and re-use

One of the first lessons we learned was that enforcement mechanisms, or any agreement process, are not as hot-swappable as we would like to think that they are. The Agreement Engine Python library supports reusing processes by “transplanting" them from existing agreement paths into new ones, allowing us to build up a library of quickly usable processes, but there are some limitations. When modeling these systems conceptually, we can abstract away from the exact data being passed around by different functions, but in order to implement the system in software, we need to explicitly state what data is being stored in an agreement, accessed in different processes, or passed to external functions. But this data resists standardization.

To see the problem, imagine a simple example where we want to swap the authoring processes of Scarce Knowledge and TSC. Scarce Knowledge receives new agreements via form submission on a website, while TSC receives new agreements by tweeting at the @agreementengine account. These two methods are both simple and have the same output types (agreements), but problems emerge if we attempt to swap them. The TSC agreement system works by reading tweets that reply to a root tweet, using it to identify unique agreement interactions, and all interactions remain on Twitter. On the other hand, Scarce Knowledge ties together a few different platforms: a simple website, Twitter direct messages, and an email plugin, and the actual content of the agreement has to be synthesized from these platforms. In other words, it is hard to design processes to be completely disentangled from the ones that come before or after them.

Here we come to one of the fundamental tradeoffs: immediate flexibility vs long-term interoperability. Currently, Agreement Engine provides useful software abstractions including the interface and path model which make it easy to set up simple agreement systems. Users don’t have to worry about the details of how the web server functions, or how to get data to specific agreements. Instead they can focus on the actual logic of the agreement, and what decisions should be made based on the incoming data. In future versions of Agreement Engine, we want to introduce similar software abstractions to path composition—i.e. other words, we

want to standardize the interfaces and possible interactions between agreement stages. From a systems perspective, we believe that the benefit of interoperability between different agreement paths is more important than preserving the short-term flexibility of being able to code whatever logic one likes.

## 5.7 Discussion

The first version of Agreement Engine was a net-native service for *registering* net-native agreements, especially on Twitter, and in a way it still serves that purpose.[3] It only evolved into a tool for implementing agreement *systems* as we tried to navigate the difficulties of creating, testing, and swapping net-native enforcement mechanisms for these agreements.

We discussed the swapping problem above. But engaging with agreement systems has also led us to a range of more social and less technical questions. Would different systems for agreement foster new solutions to persistent coordination and collective action problems? How do we reason about the tradeoffs between these systems? And by better understanding the properties and affordances of such systems, could we generate new kinds of agreements with novel properties and affordances? And on the practical side: who will be the first users of something like Agreement Engine?[4]

Our approach to engaging with these questions is empirical—through more experiments and better experiments. After all, Agreement Engine is ultimately a tool for generating and experimenting with new agreement systems. Here are some other agreement systems that we would (eventually) like to build with it:

1. A Twitter agreement, enforced by staking one's reputation on Reddit

[3]There is a subtlety here. Within any agreement path there will be a registration process that tracks whether something is an agreement. This registration may store the tracking data anywhere, including outside of the Agreement Engine. However, for practical purposes an agreement path also needs to represent and track the state of an agreement as it passes between stages. So, in effect, Agreement Engine functions as a registration service.

[4]Our hypothesis is that our first users will be communities who want to build customized contract frameworks such as Ghost Knowledge that help produce highly-specific contracts for a particular community.

2. A Markdown file with a code of conduct hosted on GitHub, enforced by a GitHub Group's owner

3. A text contract published via Google Doc, enforced by a jury of Slack users

4. A smart contract *not* on Ethereum, enforced by placing tokens in escrow on Kleros

5. A grant agreement from one DAO to another DAO, enforced by a third DAO

## 5.8 Acknowledgements

We would like to thank Lawrence Lessig, Nathan Schneider, and Miriam Ashton for helpful comments in the development of this project.

# 6
# DAOstar

How do we build an actual governance layer for online communities? This chapter extends the initial concepts and experiments developed in Chapters 4-5 into a set of real-world applications. Specifically, we introduce DAOstar, a set of open standard for organizing and publishing DAO data, and demonstrate that it satisfies the requirements articulated in Chapter 4 for a governance layer for online communities.

## Contents



## 6.1 Introduction

Decentralized autonomous organizations (DAOs) first began to surface in discussions of the blockchain community around 2013, where people imagined them as digital

substitutes for traditional organizations. Technologists argued that DAOs could automate many organizational processes and allow for more broad-based ownership and governance of the digital economy, all on the basis of a cryptographically-secured blockchain [222, 223]. While such DAOs existed mostly in the realm of speculation, with the noteworthy exception of The DAO [224], the reality of DAOs has drastically changed over the past few years. Driven by the surge of interest in crypto, real world deployment of DAOs surged by as much as 660% from 2019 to late 2020 [225]. More recently, DAOs have been deployed to fund and govern open-source infrastructure, to organize co-ops, and even to buy a copy of the US Constitution [226].

While there is no single comprehensive definition for a DAO, their core characteristics can be synthesized from various academic uses of the term: DAOs are organizations governed by a smart contract, typically deployed on a blockchain that autonomously enforces rules for interaction among the members [227]. DAOs also belong to a larger class of digitally-constituted organizations: organizations governed through computational artifacts such as software, hardware, and/or protocols. For example, the Bitcoin, Ethereum, and Tor networks, even though they are not DAOs, can be classified as digitally-constituted organizations.

In Chapter 4, we proposed a governance layer for online communities based on an open standard. In this chapter, we report on substantial work to build such a governance layer for DAOs, along with an open standard intended to organize tooling and data sets within the DAO ecosystem. In outline: we first provide some additional backgroud on DAOs and DAO science, before situating the effort to build a governance layer for DAOs within the context of their current limitations and opportunities. We then summarize the EIP-4824: Common Interfaces for DAOs standard, and report on continuing work to build DAO standards for attestations and for regulatory reporting. We conclude with a discussion about the progress and obstacles to the institutional engineering of DAOs.

## 6.2 Background

As an organizational form, DAOs are closely related to earlier forms of online community, especially open-source communities; many of the most successful DAOs are also communities organized around an open-source product. But DAOs also have antecedents in digital cooperatives and platform cooperatives [228, 229], multi-organizational networks such as keiretsus [230, 231], crowdfunding platforms such as Patreon, the economies of virtual worlds such as World of Warcraft and Second Life [232], and networked projects for peer production such as Wikipedia that question the role of the traditional firm [233]. In industry literature, DAOs are most often promoted as a technological alternative to traditional corporations [234]. Despite tremendous hype, today's DAOs are generally less competitive and less efficient at producing goods and services than comparable corporations—though there are reasons to believe that could change in the next ten years [235], some of which we reproduce in Appendix B.

As research subjects, DAOs occupy an interesting and important intersection between the social sciences and the computer and engineering sciences. DAOs are interesting because of the native availability of fine-grained data, the vibrancy of the ecosystem, a fast-moving do-ocratic culture inherited from tech and open-source, and the unique opportunities for intervention afforded by smart contracts. Compared to related fields such as platform governance or civic tech, DAO science is more economically sophisticated and invokes more cutting-edge technologies from computer science and cryptography. Compared to blockchain governance (with which it substantially overlaps), DAO science is more experimental and human-centered. Compared to older studies of online communities and virtual (game) worlds, DAO science requires significantly more legal, economic, and organizational sophistication; in DAOs, the scope of activities is broader, and the stakes higher.

Perhaps most importantly from the perspective of this chapter, the smart contracts and Ricardian contracts upon which DAOs are based are truly general-purpose instruments. That means that DAOs can express the full range of existing organizational forms—so the choice is not between a corporation and a DAO per se

but between a traditional, legally-constituted corporation and a corporation that is digitally-constituted through a smart contract. DAOs are not just curious instances of a certain kind of online community; they have the expressive potential to transport a tremendous amount of institutional infrastructure from the stonebound halls of power onto the open internet; from law and economics into computer science.

DAOs in their current form may or may not become the future of organizations. However, it is already clear that online forms of organization are becoming more and more important in the politics and economies of the world. DAO science is one of the most promising paths forward for tackling and making progress on hard questions of organization, coordination, and governance.

### 6.2.1 Current limitations

DAOs are nominally fully public and transparent, but there are many questions that researchers might like to ask which cannot be readily answered given the current data infrastructure of DAOs. For example:

- How many DAOs exist on a given blockchain at any moment?
- Who are the actual members of a DAO?
- What is the contract space of a DAO, i.e. all the on-chain contracts and multisigs controlled by or associated with a DAO?
- What is the level of engagement of members over time?
- What is the legal status of a DAO, if any?
- What is the organizational structure of a DAO, e.g. in terms of subDAOs?
- What is the spread of different voting schemes across all DAOs?
- What kinds of activities does a DAO engage in?
- What rights are afforded to the members of a DAO? Which rights are code- versus rule-enforced?

Again, the issue in these cases is not that the information is private and must be negotiated for (as with typical Web2 platforms) or that the precise definition of a metric is being contested (e.g. of decentralization). Most data relevant to DAOs is publicly-accessible, but interpreting, integrating, and organizing that data is hard. Many of these issues arise from the lack of on- and off-chain standardization across DAOs and DAO frameworks, which DAOstar attempts to address. Others arise because engaging with a DAO involves interacting with many off-chain platforms, and these platforms either do not report data or do not yet integrate with DAOstar. And while we have built some data sets for DAO frameworks (see Table 6.2.1), DAO smart contracts [236], DAO constitutions [237], and political sentiment [238], many other data sets simply still need to be built.

Despite the present lack of data infrastructure, DAOs present a unique opportunity to build "live" data sets, whether through public subgraphs and other automated mechanisms, through technical standards implemented by DAO frameworks and other service providers, or through institutional designs that incentivize reporting on a DAO-to-DAO basis. Having access to such live data is not only a boon to scientific analysis; it is also an important basis for providing real-time feedback to DAOs, drastically reducing the timeline for research to impact the real world. DAOs themselves could substantially benefit from access to such data, whether for operational, reporting, or governance purposes.

| Framework | Year | Examples | Core Features |
|---|---|---|---|
| Aragon | 2014 | Lido, Curve, Decentraland | Customization, documentation, support for enterprise |
| DAOstack / Alchemy | 2014 | dxDAO, PrimeDAO | Holographic consensus |
| Colony | 2014 | Metacolony | Lazy consensus, non-transferable voting, decaying reputation |
| Gnosis Safe | 2019 | Balancer, Constitution DAO | Multisig, app ecosystem |
| Moloch | 2019 | Moloch DAO, The LAO, Metacartel Ventures | Minimal, ragequit, non-voting shares |
| Compound Governor | 2020 | Compound, Uniswap, Gitcoin | Quorum, timelock, delegation, upgradeability |
| OpenZeppelin Governor | 2021 | ENS DAO | Customization, quorum, timelock, delegation, upgradeability |
| 1Hive Gardens | 2021 | 1Hive, BrightDAO | Participatory budgeting, dispute resolution |
| Tribute / OpenLaw | 2021 | The LAO, FlamingoDAO | Modularity, multiple tokens, guildkicks, non-voting shares |
| Sputnik (NEAR) | 2021 | Createbase | Multisig |
| Squads (Solana) | 2022 | Grape DAO | Multisig |
| Syndicate | 2022 | The Symmetrical, Outliers Fund | Legal compliance for investment clubs |
| DAODAO (Cosmos) | 2022 | RAW DAO | Multisig |

**Table 6.2.1:** Comparing DAO frameworks.

## 6.3 DAOstar and EIP-4824: Common Interfaces for DAOs

Historically, organizations have been governed according to the incorporation statutes of territorial governments; but DAOs and blockchains are often governed by technical standards [239]. Standards for DAOs will be needed to facilitate, for instance, the implementation of human rights norms, financial regulation, and the transparency necessary for corporate accountability [240].

To facilitate the development of the DAO ecosystem and to address some of the limitations raised above, we introduce DAOstar (https://daostar.org), a set of open-source technical standards, specifications, and software for DAOs and DAO tooling. At its core, DAOstar defines a standard API schema for DAO data, akin to projects such as FHIR for electronic health records or PSD2 for financial transactions. Alongside the core schema reproduced below, which is being continually refined and extended by a team of contributors, we have also deployed additional standards, applications, and infrastructure to support adoption and specific use-cases. The project includes:

- ERC-4824 / DAOIP-2: Common Interfaces for DAOs
- DAOIP-3: Attestations for DAOs
- DAOIP-4: Proposal Types
- ERC-4824 registration factories and indexing contracts deployed on many chains
- ERC-4824 reference implementations for DAO frameworks including Aragon, Snapshot, Safe, Governor, and DAODAO
- the DAOstar Endpoint Hosting Service, which supports ERC-4824 adoption

The long-term vision for DAOstar, building on the thesis of Chapter 4, is to build up DAOs as a governance layer for the internet—an interoperable ecosystem of

digital tools and social patterns that can support more effective and more equitable forms of organization than existing legal or technical infrastructure.

The sections below largely reproduce ERC-4824, an API standard for decentralized autonomous organizations focused on relating on-chain and off-chain representations of membership and proposals.

### 6.3.1 Motivation for ERC-4824

DAOs, since being invoked in the Ethereum whitepaper [241], have been vaguely defined. This has led to a wide range of patterns but little standardization or interoperability between the frameworks and tools that have emerged. Standardization and interoperability are necessary to support a variety of use-cases. In particular, a standard daoURI, similar to tokenURI in ERC-721, will enhance DAO discoverability, legibility, proposal simulation, and interoperability between tools. More consistent data across the ecosystem is also a prerequisite for future DAO standards.

DAOs, since being invoked in the Ethereum whitepaper, have been vaguely defined.

### 6.3.2 Specification

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

Every contract implementing this EIP MUST implement the `ERC4824` interface below:

```
pragma solidity ^0.8.1;

/// @title ERC-4824 DAOs
/// @dev See <https://eips.ethereum.org/EIPS/eip-4824>
interface IERC-4824 {
    event DAOURIUpdate(address daoAddress, string daoURI);

    /// @notice A distinct Uniform Resource Identifier (URI
       ) pointing to a JSON object following the "ERC-4824
      DAO JSON-LD Schema". This JSON file splits into four
        subsidiary URIs: membersURI, proposalsURI,
```

```
        activityLogURI, and governanceURI. The membersURI
        SHOULD point to a JSON file that conforms to the "
        ERC-4824 Members JSON-LD Schema". The proposalsURI
        SHOULD point to a JSON file that conforms to the "
        ERC-4824 Proposals JSON-LD Schema". The
        activityLogURI SHOULD point to a JSON file that
        conforms to the "ERC-4824 Activity Log JSON-LD
        Schema". The governanceURI SHOULD point to a
        flatfile, normatively a .md file. Each of the JSON
        files named above MAY be statically-hosted or
        dynamically-generated. The content of subsidiary
        JSON files MAY be directly embedded as a JSON object
         directly within the top-level DAO JSON, in which
        case the relevant field MUST be renamed to remove
        the "URI" suffix. For example, "membersURI" would be
         renamed to "members", "proposalsURI" would be
        renamed to "proposals", and so on.
    function daoURI() external view returns (string memory
        _daoURI);
}
```

The DAO JSON-LD Schema mentioned above:

```
{
    "@context": "http://www.daostar.org/schemas",
    "type": "DAO",
    "name": "<name of the DAO>",
    "description": "<description>",
    "membersURI": "<URI>",
    "proposalsURI": "<URI>",
    "activityLogURI": "<URI>",
    "governanceURI": "<URI>",
    "contractsURI": "<URI>"
}
```

A DAO MAY inherit the IERC-4824 interface above or it MAY create an external registration contract that is compliant with this EIP. Whether the DAO inherits the above interface or it uses an external registration contract, the DAO SHOULD define a method for and implement some access control logic to enable efficient updating for daoURI. If a DAO creates an external registration contract, the registration contract MUST store the DAO's primary address, typically the address of the primary governance contract. See the reference implementation of external registration contract in the attached assets folder to this EIP.

When reporting information in the DAO JSON-LD Schema, if a given field has no value (for example, description), it SHOULD be removed rather than left with an empty or null value.

**Indexing**

If a DAO inherits the ERC-4824 interface from a 4824-compliant DAO factory, then the DAO factory SHOULD incorporate a call to an indexer contract as part of the DAO's initialization to enable efficient network indexing. If the DAO is ERC-165-compliant, the factory can do this without additional permissions. If the DAO is not compliant with ERC-165, the factory SHOULD first obtain access control rights to the indexer contract and then call logRegistration directly with the address of the new DAO and the daoURI of the new DAO. Note that any user, including the DAO itself, MAY call logRegistration and submit a registration for a DAO which inherits the ERC-4824 interface and which is also ERC-165-compliant.

```
pragma solidity ^0.8.1;

error ERC-4824InterfaceNotSupported();

contract ERC-4824Index is AccessControl {
    using ERC165Checker for address;

    bytes32 public constant REGISTRATION_ROLE = keccak256("
       REGISTRATION_ROLE");

    event DAOURIRegistered(address daoAddress);

    constructor() {
        _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
        _grantRole(REGISTRATION_ROLE, msg.sender);
    }

    function logRegistrationPermissioned(
        address daoAddress
    ) external onlyRole(REGISTRATION_ROLE) {
        emit DAOURIRegistered(daoAddress);
    }

    function logRegistration(address daoAddress) external {
```

```
        if (!daoAddress.supportsInterface(type(IERC-4824).
            interfaceId))
            revert ERC-4824InterfaceNotSupported();
        emit DAOURIRegistered(daoAddress);
    }
}
```

If a DAO uses an external registration contract, the DAO SHOULD use a common registration factory contract linked to a common indexer to enable efficient network indexing. See the reference implementation of the factory contract in the attached assets folder to this EIP.

**Indexing priority.** daoURIs may be published directly in the DAO's contract or through a call to a common registration factory contract. In cases where both occur, the daoURI (and all sub-URIs) published through a call to a registration factory contract SHOULD take precedence. If there are multiple registrations, the most recent registration SHOULD take precedence.

### Members

Members JSON-LD Schema. Every contract implementing this EIP SHOULD implement a membersURI pointing to a JSON object satisfying this schema.

```
{
    "@context": "https://www.daostar.org/schemas",
    "type": "DAO",
    "members": [
        {
            "id": "<CAIP-10 address, DID address, or other
                URI identifier>"
        },
        {
            "id": "<CAIP-10 address, DID address, or other
                URI identifier>"
        }
    ]
}
```

For example, for an address on Ethereum Mainnet, the CAIP-10 address would be of the form eip155:1:0x1234abcd, while the DID address would be of the form did:ethr:0x1234abcd.

### Proposals

Proposals JSON-LD Schema. Every contract implementing this EIP SHOULD implement a proposalsURI pointing to a JSON object satisfying this schema.

In particular, any on-chain proposal MUST be associated to an id of the form CAIP10_ADDRESS + "?proposalId=" + PROPOSAL_COUNTER, where CAIP10_ADDRESS is an address following the CAIP-10 standard and PROPOSAL_COUNTER is an arbitrary identifier such as a uint256 counter or a hash that is locally unique per CAIP-10 address. Off-chain proposals MAY use a similar id format where CAIP10_ADDRESS is replaced with an appropriate URI or URL.

```
{
    "@context": "https://www.daostar.org/schemas",
    "proposals": [
        {
            "type": "proposal",
            "id": "<proposal ID>",
            "name": "<name or title of proposal>",
            "contentURI": "<URI to content/text of the
              proposal>",
            "discussionURI": "<URI to discussion or thread
              for the proposal>",
            "status": "<status of proposal>",
            "calls": [
                {
                    "type": "CallDataEVM",
                    "operation": "<call or delegate call>",
                    "from": "<EthereumAddress>",
                    "to": "<EthereumAddress>",
                    "value": "<value>",
                    "data": "<call data>"
                }
            ]
        }
    ]
}
```

When deferenced, contentURI should return the content (i.e. the text) of the proposal. Similarly, discussionURI should return a discussion link, whether a forum post, Discord channel, or Twitter thread.

**Activity Log**

Activity Log JSON-LD Schema. Every contract implementing this EIP SHOULD implement a activityLogURI pointing to a JSON object satisfying this schema.

```
{
    "@context": "https://www.daostar.org/schemas",
    "activities": [
        {
            "id": "<activity ID>",
            "type": "activity",
            "proposal": {
                "type": "proposal"
                "id": "<proposal ID>",
            },
            "member": {
                "id": "<CAIP-10 address, DID address, or
                   other URI identifier>"
            }
        }
    ]
}
```

**Contracts**

Contracts JSON-LD Schema. Every contract implementing this EIP SHOULD implement a contractsURI pointing to a JSON object satisfying this schema.

contractsURI is especially important for DAOs with distinct or decentralized governance occurring across multiple different contracts, possibly across several chains. Multiple addresses may report the same daoURI.

To prevent spam or spoofing, all DAOs adopting this specification SHOULD publish through contractsURI the address of every contract associated to the DAO, including but not limited to those that inherit the ERC-4824 interface or those that interact with an ERC-4824 registration factory contract. Note that this includes the contract address(es) of any actual registration contracts deployed through a registration factory.

```
{
    "@context": "https://www.daostar.org/schemas",
    "contracts": [
```

```
        {
            "id": "<CAIP-10 address, DID address, or other
              URI identifier>"
            "name": "<name, e.g. Treasury>",
            "description": "<description, e.g. Primary
              operating treasury for the DAO>"
        },
        {
            "id": "<CAIP-10 address, DID address, or other
              URI identifier>"
            "name": "<name, e.g. Governance Token>",
            "description": "<description, e.g. ERC20
              governance token contract>"
        },
        {
            "id": "<CAIP-10 address, DID address, or other
              URI identifier>"
            "name": "<name, e.g. Registration Contract>",
            "description": "<description, e.g. ERC-4824
              registration contract>"
        }
    ]
}
```

**URI fields**

The content of subsidiary JSON files MAY be directly embedded as a JSON object directly within the top-level DAO JSON, in which case the relevant field MUST be renamed to remove the "URI" suffix. For example, membersURI would be renamed to members, proposalsURI would be renamed to proposals, and so on. In all cases, the embedded JSON object MUST conform to the relevant schema. A given field and a URI-suffixed field (e.g. membersURI and members) SHOULD NOT appear in the same JSON-LD; if they do, the field without the URI suffix MUST take precedence.

Fields which are not appended with URI MAY be appended with a URI, for example name and description may be renamed to nameURI and descriptionURI, in which case the dereferenced URI MUST return a JSON-LD object containing the "@context": "https://www.daostar.org/schemas" field and the original key-value pair.

For example, descriptionURI should return:

```
{
    "@context": "https://www.daostar.org/schemas",
    "description": "<description>"
}
```

**Entities which are not DAOs**

Entities which are not DAOs or which do not wish to identify as DAOs MAY still publish daoURIs. If so, they SHOULD use a different value for the type field than "DAO", for example "Organization", "Foundation", "Person", or, most broadly, "Entity".

Entities which are not DAOs or which do not wish to identify as DAOs MAY also publish metadata information through an off-chain orgURI or entityURI, which are aliases of daoURI. If these entities are reporting their URI through an on-chain smart contract or registration, however, they MUST retain the ERC-4824 daoURI interface in order to enable network indexing.

The Entity JSON-LD Schema:

```
{
    "@context": "https://www.daostar.org/schemas",
    "type": "<type of entity>",
    "name": "<name of the entity>",
    "description": "<description>",
    "membersURI": "<URI>",
    "proposalsURI": "<URI>",
    "activityLogURI": "<URI>",
    "governanceURI": "<URI>",
    "contractsURI": "<URI>"
}
```

### 6.3.3 Rationale

In this standard, we assume that all DAOs possess at least two primitives: *membership* and *behavior*. *Membership* is defined by a set of addresses. *Behavior* is defined by a set of possible contract actions, including calls to external contracts and calls to internal functions. *Proposals* relate membership and behavior; they are objects that members can interact with and which, if and when executed, become behaviors of the DAO.

### APIs, URIs, and off-chain data

DAOs themselves have a number of existing and emerging use-cases. But almost all DAOs need to publish data off-chain for a number of reasons: communicating to and recruiting members, coordinating activities, powering user interfaces and governance applications such as Snapshot or Tally, or enabling search and discovery via platforms like DeepDAO, Messari, and Etherscan. Having a standardized schema for this data organized across multiple URIs, i.e. an API specification, would strengthen existing use-cases for DAOs, help scale tooling and frameworks across the ecosystem, and build support for additional forms of interoperability.

While we considered standardizing on-chain aspects of DAOs in this standard, particularly on-chain proposal objects and proposal IDs, we felt that this level of standardization was premature given (1) the relative immaturity of use-cases, such as multi-DAO proposals or master-minion contracts, that would benefit from such standardization, (2) the close linkage between proposal systems and governance, which we did not want to standardize (see "governanceURI", below), and (3) the prevalence of off-chain and L2 voting and proposal systems in DAOs (see "proposalsURI", below). Further, a standard URI interface is relatively easy to adopt and has been actively demanded by frameworks (see "Community Consensus", below).

We added the ability to append or remove the URI suffix to make dereferenced daoURIs easier to parse, especially in certain applications that did not want to maintain several services or flatfiles. Where there is a conflict, we decided that fields without the URI suffix should take precedence since they are more directly connected to the initial publication of daoURI.

In terms of indexing: we believe that the most trustworthy way of publishing a daoURI is through an on-chain registration contract, as it is the clearest reflection of the active will of a DAO. It is also the primary way a DAO may "overwrite" any other daoURI that has previously been published, through any means. If a DAO inherits daoURI directly through its contract, then this information is also

trustworthy, though slightly less so as it often reflects the decisions of a DAO framework rather than the DAO directly.

**membersURI**

Approaches to membership vary widely in DAOs. Some DAOs and DAO frameworks (e.g. Gnosis Safe, Tribute), maintain an explicit, on-chain set of members, sometimes called owners or stewards. But many DAOs are structured so that membership status is based on the ownership of a token or tokens (e.g. Moloch, Compound, DAOstack, 1Hive Gardens). In these DAOs, computing the list of current members typically requires some form of off-chain indexing of events.

In choosing to ask only for an (off-chain) JSON schema of members, we are trading off some on-chain functionality for more flexibility and efficiency. We expect different DAOs to use membersURI in different ways: to serve a static copy of on-chain membership data, to contextualize the on-chain data (e.g. many Gnosis Safe stewards would not say that they are the only members of the DAO), to serve consistent membership for a DAO composed of multiple contracts, or to point at an external service that computes the list, among many other possibilities. We also expect many DAO frameworks to offer a standard endpoint that computes this JSON file, and we provide a few examples of such endpoints in the implementation section.

We encourage extensions of the Membership JSON-LD Schema, e.g. for DAOs that wish to create a state variable that captures active/inactive status or different membership levels.

**proposalsURI**

Proposals have become a standard way for the members of a DAO to trigger on-chain actions, e.g. sending out tokens as part of a grant or executing arbitrary code in an external contract. In practice, however, many DAOs are governed by off-chain decision-making systems on platforms such as Discourse, Discord, or Snapshot, where off-chain proposals may function as signaling mechanisms for an administrator or as a prerequisite for a later on-chain vote. (To be clear, on-chain votes may also serve as non-binding signaling mechanisms or as “binding” signals leading to some

sort of off-chain execution.) The schema we propose is intended to support both on-chain and off-chain proposals, though DAOs themselves may choose to report only on-chain, only off-chain, or some custom mix of proposal types.

**Proposal ID**. In the specification, we state that every unique on-chain proposal must be associated to a proposal ID of the form CAIP10_ADDRESS + "?proposalId=" + PROPOSAL_COUNTER, where PROPOSAL_COUNTER is an arbitrary string which is unique per CAIP10_ADDRESS. Note that PROPOSAL_COUNTER may not be the same as the on-chain representation of the proposal; however, each PROPOSAL_COUNTER should be unique per CAIP10_ADDRESS, such that the proposal ID is a globally unique identifier. We endorse the CAIP-10 standard to support multi-chain / layer 2 proposals and the "?proposalId=" query syntax to suggest off-chain usage.

**ContentURI**. In many cases, a proposal will have some (off-chain) content such as a forum post or a description on a voting platform which predates or accompanies the actual proposal.

**Status**. Almost all proposals have a status or state, but the actual status is tied to the governance system, and there is no clear consensus between existing DAOs about what those statuses should be (see table below). Therefore, we have defined a "status" property with a generic, free text description field.

| **Project** | **Proposal States** |
|---|---|
| Aragon | Not specified |
| Colony | ['Null', 'Staking', 'Submit', 'Reveal', 'Closed', 'Finalizable', 'Finalized', 'Failed'] |
| Compound | ['Pending', 'Active', 'Canceled', 'Defeated', 'Succeeded', 'Queued', 'Expired', 'Executed'] |
| DAOstack/ Alchemy | ['None', 'ExpiredInQueue', 'Executed', 'Queued', 'PreBoosted', 'Boosted', 'QuietEndingPeriod'] |
| Moloch v2 | [sponsored, processed, didPass, cancelled, whitelist, guildkick] |
| Tribute | ['EXISTS', 'SPONSORED', 'PROCESSED'] |

**Table 6.3.1:** Different states that a proposal can be in within different DAO frameworks.

**ExecutionData**. For on-chain proposals with non-empty execution, we include an array field to expose the call data. The main use-case for this data is execution simulation of proposals.

**activityLogURI**

The activity log JSON is intended to capture the interplay between a member of a DAO and a given proposal. Examples of activities include the creation/submission of a proposal, voting on a proposal, disputing a proposal, and so on.

*Alternatives we considered: history, interactions*

**governanceURI**

Membership, to be meaningful, usually implies rights and affordances of some sort, e.g. the right to vote on proposals, the right to ragequit[1], the right to veto proposals, and so on. But many rights and affordances of membership are realized off-chain (e.g. right to vote on a Snapshot, gated access to a Discord). Instead of trying to standardize these wide-ranging practices or forcing DAOs to locate descriptions of those rights on-chain, we believe that a flatfile represents the easiest and most widely-acceptable mechanism for communicating what membership means and how proposals work. These flatfiles can then be consumed by services such as Etherscan, supporting DAO discoverability and legibility.

We chose the word "governance" as an appropriate word that reflects (1) the widespread use of the word in the DAO ecosystem and (2) the common practice of emitting a governance.md file in open-source software projects.

*Alternative names considered: description, readme, constitution*

**contractsURI**

Over the course of community conversations, multiple parties raised the need to report on, audit, and index the different contracts belonging to a given DAO. Some of these contracts are deployed as part of the modular design of a single DAO

[1]Ragequit is a term originating from MolochDAO, indicating a right to withdraw a portion of deposited funds when leaving the DAO.

framework, e.g. the core, voting, and timelock contracts within Open Zeppelin / Compound Governor. In other cases, a DAO might deploy multiple multsigs as treasuries and/or multiple subDAOs that are effectively controlled by the DAO. contractsURI offers a generic way of declaring these many instruments so that they can be efficiently aggregated by an indexer.

contractsURI is also important for spam prevention or spoofing. Some DAOs may spread governance power and control across multiple different governance contracts, possibly across several chains. To capture this reality, multiple addresses may wish to report the same daoURI, or different daoURIs with the same name. However, unauthorized addresses may try to report the same daoURI or name. Additional contract information can prevent attacks of this sort by allowing indexers to weed out spam information.

*Alternative names considered: contractsRegistry, contractsList*

**Why JSON-LD**

We chose to use JSON-LD rather than the more widespread and simpler JSON standard because (1) we want to support use-cases where a DAO wants to include members using some other form of identification than their Ethereum address and (2) we want this standard to be compatible with future multi-chain standards. Either use-case would require us to implement a context and type for addresses, which is already implemented in JSON-LD.

Further, given the emergence of patterns such as subDAOs and DAOs of DAOs in large organizations such as Synthetix, as well as L2 and multi-chain use-cases, we expect some organizations will point multiple DAOs to the same URI, which would then serve as a gateway to data from multiple contracts and services. The choice of JSON-LD allows for easier extension and management of that data.

**Community Consensus**

The initial draft standard was developed as part of the DAOstar roundtable series, which included representatives from all major EVM-based DAO frameworks (Aragon, Compound, DAOstack, Gnosis, Moloch, OpenZeppelin, and Tribute), a

wide selection of DAO tooling developers, as well as several major DAOs. Thank you to all the participants of the roundtable. We would especially like to thank Fabien of Snapshot, Jake Hartnell, Auryn Macmillan, Selim Imoberdorf, Lucia Korpas, and Mehdi Salehi for their contributions.

In-person events for community comment were held at Schelling Point 2022, ETHDenver 2022, ETHDenver 2023, DAO Harvard 2023, DAO Stanford 2023 (also known as the Science of Blockchains Conference DAO Workshop). We have also hosted over 50 biweekly community calls as part of the DAOstar project.

### 6.3.4 Backwards Compatibility

Existing contracts that do not wish to use this specification are unaffected. DAOs that wish to adopt the standard without updating or migrating contracts can do so via an external registration contract.

### 6.3.5 Security Considerations

This standard defines the interfaces for the DAO URIs but does not specify the rules under which the URIs are set, or how the data is prepared. Developers implementing this standard should consider how to update this data in a way aligned with the DAO's governance model, and keep the data fresh in a way that minimizes reliance on centralized service providers.

Indexers that rely on the data returned by the URI should take caution if DAOs return executable code from the URIs. This executable code might be intended to get the freshest information on membership, proposals, and activity log, but it could also be used to run unrelated tasks.

## 6.4 Discussion

In Chapter 4 we introduced four desired properties for a governance layer for online communities:

1. **Modularity**: Platform operators and community members should have the ability to construct systems by creating, importing, and arranging composable parts together as a coherent whole.

2. **Expressiveness**: The governance layer should be able to implement as wide a range of processes as possible.

3. **Portability**: Governance tools developed for one platform should be portable to another platform for reuse and adaptation.

4. **Interoperability**: Governance systems operating on different platforms and protocols should have the ability to interact with each other, sharing data and influencing each other's processes.

We claim that DAOstar, along with the constellation of DAOstar-compliant DAOs and services, constitutes such a governance layer.

1. **Modularity**: At a services level, modularity is built into the architecture of DAOstar as an API standard; DAOs can connect to many independent services (for voting, for treasury management, for reputation, etc.) that together define the operation of the DAO. At the smart contract level, many DAO frameworks such as Governor, Safe, and Aragon have independently moved to more modular design principles that offload different aspects of governance (timelock, permissions, oracles) into separate smart contracts.

2. **Expressiveness**: We have already argued above for the native expressiveness of DAOs and smart contracts for generating a wide range of institutional forms, from corporations to cooperatives to open-source communities. This is part of what makes them interesting.

3. **Portability**: DAOstar directly supports the portability and reuse of DAO tooling across different DAOs and DAO frameworks; EIP-4824 was motivated directly by the desire of frameworks to allow DAO tooling to work across multiple frameworks, so that tooling developers did not have to learn multiple frameworks, or commit to just one framework, in building new tools.

4. **Interoperability**: Blockchains natively allow the sharing of data; DAOs can also natively transfer money and assets to other DAOs on the same blockchain; DAOstar standardizes DAO data to enable richer interactions between DAOs as well as better tooling. For example, the DAOstar attestations standard supports the Web3 profiles and CVs, allowing DAOs to independently report contributions and user information that can then be aggregated across many different services and platforms. At a smart contract level, many frameworks are already closely related to each other; for example, Moloch v2 DAOs, via their Minions framework, can effectively own and govern a Gnosis Safe. 1Hive Gardens is a community-directed direct fork of Aragon's original contracts, while Open Zeppelin Governor is a fork of Compound Governor that highlights Open Zeppelin's work auditing and securing contracts. We are working to define additional contract-level standards to facilitate more interesting forms of DAO-to-DAO interaction.

## 6.5 Conclusion

In Chapter 1, we described a broad research program to engineer institutions that draws on both centuries-old legal traditions as well as Ostrom's legacy of institutional analysis and design (IAD). DAOs mark a significant step in this program. As programmable constructs, DAOs and smart contracts allow us to convert many long-standing social, organizational, and legal problems into computational ones, opening up interesting new challenges for computer scientists, mathematicians, and engineers. As explored above, their decentralized and open-source nature empowers individuals to create, deploy, and experiment with a vast range of institution structures from the bottom-up, in some ways fulfilling the promise of IAD. And while DAOs cannot yet express *all* organizational forms, technological advances in cryptography, human-computer interaction, and AI (see Chapter 7) continue to widen their feature sets, including more granular privacy primitives, decentralized jury systems, and interoperable regulatory systems.

Much work remains. While DAOs have various existing and theorized benefits compared to traditional corporations (as covered in Appendix B), scalability and security issues still dog the ecosystem; while performant, DAOs are still substantially less forgiving of user mistakes than traditional bank accounts. Regulatory uncertainty with respect to legal personhood and limited liability also sharply constrains DAO adoption—something DAOstar is already working to address through *regulatory interoperability*, a project to standardize DAO communication with state registrars and to help DAOs understand, audit, and obtain legal benefits such as limited liability. Perhaps most importantly, significant work remains to be done to decrease the costs of coordination within DAOs so that they are economically advantageous compared to traditional corporations.

# 7
# Conclusion

## Contents



In January 1956, the political scientist Herbert Simon and his former student Allen Newell gathered together a small group of people—"my wife and three children together with some graduate students," said Simon [242]—and gave to each of those people a 3-by-5 inch card. Simon and Newell then taught them to play a simple game. Each person in the game, upon receiving a message from someone, would follow the instructions on their card and relay a new message to another person in the room. The game of message-passing would go on until one of the cards stated that the game was over. The resulting performance was the first run of what is arguably the first artificial intelligence program: the Logic Theorist [243].

By 1956, Simon was already famous for his theories of bureaucracy and the administrative state [244]. The Logic Theorist was typical of bureaucratic decision-making, and it could "think" in the way that bureaucracies could think: through explicit and routine processes. Vice versa, social institutions such as bureaucracies can also constitute a model of computation insofar as they can follow instructions to the letter. "The organization of a complex job whether it is done by human

clerks, by punched cards, or by high-speed computers is bound to be a long business, and a program is only a coded form of this organization" [242, p. 421]. Of course, a bureaucracy is an extreme form of social institution, just as the Logic Theorist is now considered an extreme form of AI.

Thirty years later, another founder of AI, Marvin Minsky, made a sharp delineation between those "mindless" processes called agents and the greater AI which they might give rise to, what he called a "society of mind" [245]. How would we create such a society? Minsky did not say. And despite the analogy, Minsky was not really thinking of designing AI as if it were a human society, governed by human institutions.[1]

Today, people in AI tend to refer to the Logic Theorist as "good-old-fashioned AI", meaning old-fashioned compared to "modern AI".[2] The preeminent textbook in the field (literally, *Artificial Intelligence: A Modern Approach*) defines modern AI as the study of rational agents, where an agent is something that perceives and acts and a rational agent is one which acts "so as to achieve one's goals, given one's beliefs", with the caveat that, in AI, one restricts oneself to experiments which can be conducted on computers rather than on people [247].[3] This simple idea—that AI is about computational agents that act to achieve a goal—succeeded in unifying and organizing the entire field, from the good old-fashioned Logic Theorist to statistical learning algorithms like deep neural networks to profoundly un-simple robots like the Google Car.[4]

[1] A human society is, crucially, a society of *minds*.

[2] It's an unfortunate comparison—unfortunate because it gives people the wrong impression of modern AI. People teach AI these days as if modern AI surpassed old-fashioned AI merely because old robots like Shakey [246] couldn't deal with the real world. That's not wrong, but it misses the core conceptual distinction between a specific class of decision-making algorithms—good-old-fashioned logical methods—and a broader framework for understanding agents. Modern AI is not modern because it surpassed older AI techniques. It is modern because it defines "AI" in a very specific way, relative to a task or an objective function.

[3] Notably, a rational agent is not necessarily something that thinks logically, by means of logical implication.

[4] "I honestly don't know why it became so popular. What we tried to do was present AI as a unified discipline, and that hadn't been done before in textbooks. Previously, AI textbooks had similar chapter titles to the ones we had, but there was no attempt to tie together search, planning, reasoning under uncertainty, language. These were all just separate disciplines. And what we tried to do was say, really it was all one thing, which was how do you convert your sensory experience over your lifetime, to behavior, and that could be verbal behavior, that could be the

Why has the rational agent approach worked so well for AI? In part: (1) because all working examples of AI have been task-specific, not general-purpose, (2) because tasks (and domains) in AI are constructed so that they can be well-modeled and summarized by a single objective function, and (3) because this pairing of an objective function with the data that defines the task leads to a very practical and efficient way of defining the solution space.[5] But what if any of these assumptions breaks down? What if we want AI to be general-purpose; what if tasks can no longer be modeled by single objective functions; what if the task or solution space is unknown, vague, or dynamic?

## 7.1 From collective to artificial intelligence

Since the 1990s, we have seen a growing number of practical algorithms in AI that work by organizing the efforts of many agents to a collective end, from ensemble methods that combine the inputs of many classifiers to differentiable games that match machines against each other and against themselves.[6] In the past decade, research in machine learning has gravitated toward such techniques due to their spectacular performance in unsupervised image and natural language tasks. A crucial feature of these methods is the way in which agents interact, and the degree to which that interaction is tailored through the careful composition of loss functions. There are many examples of this: generative adversarial networks (GANs) [249], adversarial training [250], intrinsic curiosity modules for reinforcement learning [251], warped gradients for meta-learning [252], hyperparameter optimization through implicit differentiation [253], and so on. Even older techniques like AdaBoost

physical behavior in a robot, it could be just displaying a chess move on a screen, and these are all instances of the same, underlying problem" [248].

[5]Compare this with some of the complaints about rational choice theory: it fails (1) because humans in the wild are embedded within complex socioeconomic institutions, not single tasks, (2a) because these institutions are hard to construct and hard to test, (2b) because humans do not optimize a single utility function within any institution, or even within a single task, (2c) because institutions do not, generally speaking, have single objective functions, and (3) because the theories built on top of rational choice—including game theory, mechanism design, and social choice—actually provide very little guidance on how to design heterogeneous institutions.

[6]We also reference various multi-agent systems, though many multi-agent systems are used for simulation or as settings for testing single-agent algorithms, not in order to achieve a collective end.

[254] feature a "total" loss objective which is a composite of the one used in its base classifier. These examples all feature fine-tuned interactions between different objectives, but defining these interactions tends to be ad hoc and time-intensive.

And much like the Logic Theorist or Minsky's society of mind, many of these algorithms are reminiscent of extant institutional patterns in economics and the social sciences. However, there has been little work to articulate or build on that connection, though there have been calls for an "artificial social engineering" for differentiable games as recently as 2019 [255].

Earlier versions of this thesis began with the hope that we could cycle the idea of an institution from social science into AI, and then back again from AI into social science. Doing so would help us carry over intuitions, concepts, and even theorems from one to the other, and it would help us understand the opportunities and limitations of compositionality. I still believe that this project is possible and needful. Indeed, many objects in AI can be productively understood as institutions, e.g. differentiable games such as GANs, which are institutions where the agents are interacting neural networks, or end-to-end learning algorithms, where the agents are the neurons, layers, or submodules of a deep neural network [256, 257].

More concretely, an institutional perspective on AI could help us invent new ways of reasoning about differentiable games and better understand the kinds of statements and guarantees that we can make based solely on the architecture of interactions between deep networks, rather than derive generalization bounds based on the modules or network architecture of a single network, a distinction that I like to think of as "topological" versus "geometric".[7] Institutions can also help us think about the relationships and interactions between different pre-trained models, a critical question in the emerging world of many models.

[7]Topology tends to study properties of spaces under continuous deformation, while geometry studies more "rigid" features of a space, e.g. distances under a particular metric. For example, mathematicians often speak of topological invariants as coarser than geometric invariants, since they don't detect as many changes to the space. By analogy, an institutional, "topological" perspective on AI focuses on reasoning about the architecture of interactions between several networks, while a traditional, "geometric" perspective focuses on the specific, quantifiable details of a single network.

## 7.2 From artificial to collective intelligence

In this thesis, I have tried to contribute to the foundations of institutional engineering, inspired by legal practice, by institutional analysis and design, and by modern software engineering. But legal practice, IAD, and software engineering all share a certain way of looking at the world, one that imposes certain constraints.

Consider the fact that everything in a DAO is written out explicitly in the form of smart contract code. This means that when a large DAO like Uniswap votes on a governance parameter, they are really buying into a very explicit representation of their organization and how to interact with it. When a DAO like Gitcoin deploys a governance token, they are defining a very explicit form of power in the organization. This explicitness has consequences. As DAOs have discovered, it is hard to change these parameters, or at least no less hard than it is in legal organizations. As Margaret Hagan once said, "It is more fun to craft these visual and human parts of policy. It is exhausting and contentious to get text exactly right" [258].

A DAO, in other words, is a *symbolic organization*: an organization with an explicit, human-readable representation or description.

The Logic Theorist was a prime example of what we call *symbolic AI*, which is an AI that represents the world through explicit, human-readable symbols and variables. Just to be extra clear: a symbolic AI represents the world through symbols that *humans* can read and interpret.[8] Examples in AI include logic programming, search, and various forms of causal inference. Most programming languages and software engineering practices are also very symbolic in flavor.

In contrast, a *subsymbolic AI* is an AI that does not seek to represent the world through explicit symbols and variables. Subsymbolic AI are very popular these days. Examples include robots like Genghis that emphasize interacting directly with the world, statistical learning methods that represent the world through large,

[8] It's important to understand that symbolic and subsymbolic, despite being nominally about symbols and formal variables, contain very little mathematical or technical content. No one has defined them precisely. They are very old concepts that talk about the relationship between an AI and the people who build and maintain it.

unstructured data sets, and neural networks that encode features of the world within billions of weight parameters.

This suggests the following definition: a *subsymbolic organization* is an organization without an explicit, human-readable representation or description. So no text. No contracts. No org chart. No explicit rules. But not necessarily no rules.

Consider the Facebook Newsfeed. Or the platform formerly known as Twitter. These are interfaces, often powered by complicated AI underneath the hood, that mediate your interactions with a set of other people. They impose rules and conditions on those interactions; we may be aware of some of those rules and conditions, but the vast majority we'll never know. These are examples of subsymbolic organizations.

This leads us to the crux of the issue. Open games from Chapter 2 and Chapter 3, Orgs and Platforms from Chapter 4, digital agreements from Chapter 5, and DAOs from Chapter 6 *are all symbolic*. Almost every example in this thesis revolves around an explicit representation, often with the goal making it more accessible (i.e. more human-readable!) or making it easier to build up ever bigger—but no less explicit—representations. Compositionality, which tells us how build things of one type out of things of the same type, requires an explicit representation of type; of structure. I believe that these methods represent a significant improvement on existing methods for institutional design, but I also recognize that they are so, so far from being able to compete with the speed and power of modern machine learning. If we want to advance social science as quickly as we have advanced AI in the past 30 years, then I believe we need to move away from the symbolic approach. Maybe symbolic is bad. Maybe we don't need to expose all these parameters. Maybe we can engineer and run better institutions, faster, if we got rid of all the technical representations that only a few specialists will ever read or understand.[9]

Earlier, I talked about Facebook Newsfeed and Twitter. These are examples of subsymbolic organizations—sort of. Sort of because these interfaces do not really

[9]More likely than not, we need to find ways for symbolic and subsymbolic methods in institutional design to complement each other. But to do that, we need to start building subsymbolic organizations!

encode "organizational structure" into their internal models. But what if they did? What if LinkedIn recommended tasks and matched your daily output into a set of collective decisions? What if GitHub replaced its entire permissions and role structure with an open-ended statistical method for merging pull requests? What if Notion or Slack had a neural network that could generate a website on the fly that reflected the practices, structures, and incentives of an organization? What if we, as founders, as scientists, or as lawyers could intervene on an organization not by trying to change its rules and processes but by intervening directly on an interface, on an algorithm, or on a data set?

I'm not sure what will happen. These experiments have yet to be built.

# Appendices

# A

# Modeling procedures for Clause2Game

## Contents



1. In each example, we begin with an existing legal contract on Lawgood, meaning that:

   (a) As in a traditional contract, we have a written document broken into sections and clauses. A clause typically appears as the whole text of a section or as the whole text of a named subsection within a longer section.

   (b) Some clauses are “highlighted" in that they can be swapped out for other, related clauses. The related clauses are provided next to the original, default clause text as part of the overall contract document.

   (c) In addition, some parameters are highlighted within the contract. Some of these are “strategic" e.g. the amount of a late fee, the percentage ownership over an asset, or whether some process is required or not for a

clause to apply. Other parameters are “non-strategic" within the context of our model, e.g. the signature date or the name of one of the parties.

2. For each contract, we generate a document model of that contract, including all sections, highlighted clauses, and highlighted parameters. We mark down parameters that are non-strategic and exclude those from our analysis.

3. For each clause, we associate it to a clause category—note that as a clause category is *not* a category in the mathematical sense, but a specific family of open games. Often, but not always, this clause category corresponds to the section title associated to that clause.

4. For each clause category, we model it as a generic open game, in particular specifying the minimal input, feedback, output, and return variables expected of all clauses within that clause category along with their types. This allows us to verify and automate various compositions.

5. For each clause, we produce a set of closed games as models of those clauses. The closed games are typically built out of open components which can be reused across other clause models in that clause category. For certain clause categories, we model the differences between clauses through a simple cost function.

6. We produce a set of strategies for every abstract model. These strategies can be thought of as tests or queries to the model—they are ways of “running" the model for a specific set of agent behaviors and checking whether those behaviors constitute an equilibrium of some sort.

7. Finally, we export everything to a relationship database composed of six main tables: Contracts, Clauses, Clause Categories, Games, Variables, and Strategies.

For more details, see the repository at `https://github.com/thelastjosh/clause2game`.

## A.1 Data structures for compositional modeling

In compositional modeling, there is a tradeoff between modeling a clause category through a single model (i.e. a cost function) versus through modeling separately each of the clauses / options in the category. The former is simpler, more straightforward, and more in line with typical software engineering, which tries to remove redundancy. The latter is more granular, and allows us to easily substitute / modify a given clause. In this case, we decided that the more granular and potentially-redundant option made more sense. This is because (1) granularity and flexibility is a significant advantage, (2) each clause represents a “fork" (in software engineering parlance) of the whole contract, so concerns about redundancy do not apply in the same way, and (3) we want to reserve the text as the ground truth of these contracts, which motivates us to preserve even minimal differences between clauses.

We did end up creating a separate data structure to represent parameters in a contract, along with some constraints on how parameters compose with clause categories to generate a space of contracts or perform automated testing on this data structure.

## A.2 Additional tables and diagrams for Clause2Game

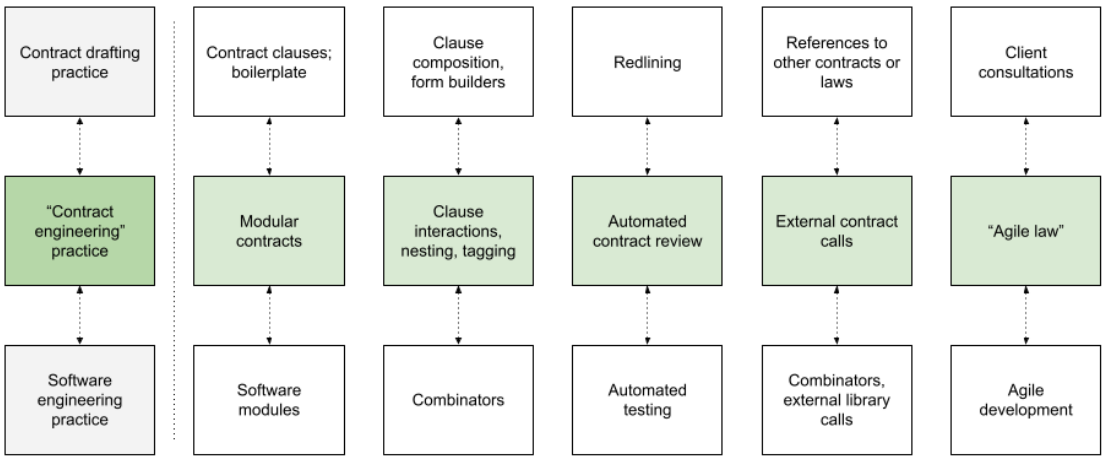



**Figure A.2.1:** Triangulating the practices of contract engineering from contract drafting practices and software engineering practices.

| Section | Parameters or options |
|---|---|
| Price | The Price does not include, and the Purchaser shall be fully responsible for, any [transportation costs, freight, insurance, special handling and packaging, or any required federal, state, or local sales or other taxes, duties, export or custom charges, VAT charges, brokerage, or other fees]; provided, however, that the Purchaser shall not be responsible for any taxes imposed on, or with respect to, the Seller's income, revenues, gross receipts, personnel, real or personal property, or other assets. |
| Payment | The Purchaser shall pay all invoiced amounts due to the Seller [on receipt][within NUMBER days from the date] of the Seller's invoice. The Purchaser shall make all payments hereunder in U.S. dollars [by cash, check, or wire transfer]. The Purchaser shall pay interest on all late payments, at the lesser of the annual interest rate of [one and a half percent (1.5%)] or the highest rate permissible under applicable law. [In addition to all other remedies available under this Agreement or at law (which the Seller does not waive by the exercise of any rights under this Agreement), if the Purchaser fails to pay any amounts when due under this Agreement, the Seller may (i) suspend the delivery of any Goods or (ii) terminate this Agreement pursuant to the terms of Section 10 (Termination).] |
| Delivery Location | Shipper's Location OR Purchaser's Location |
| Quantity | The Seller shall have the option of shipping up to [15]% more or less than the quantity set forth in Exhibit A. |
| Packaging and Shipping | Favorable to Purchaser OR Neutral OR Favorable to Seller OR Very Favorable to Seller |
| Title and Risk of Loss | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Inspection | Favorable to Purchaser OR Favorable to Seller |
| Product Warranties | Favorable to Purchaser OR Neutral (1) OR Neutral (2) OR Favorable to Seller |

**Table A.2.1:** All clauses, parameters, and options from Lawgood's sale of goods contract.

| **Section** | **Parameters or options** |
|---|---|
| Intellectual Property | N/A |
| Insurance | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Indemnification | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Limitation of Liability | Neutral OR Favorable to Seller |
| Termination by the Seller | Favorable to Purchaser OR Neutral OR Favorable to Seller OR Most Favorable to Seller |
| Termination by the Purchaser | Favorable to Purchaser OR Neutral OR Favorable to Seller |
| Effect of Expiration or Termination | The termination of this Agreement will not affect any rights or obligations of the Parties that: (i) come into effect upon or after the termination of this Agreement or (ii) otherwise survive the termination of this Agreement in accordance with its terms or were incurred by the Parties prior to such termination[, including the Purchaser's obligation to pay for all Goods delivered up to the date of termination]. |
| Governing Law | This Agreement and all related documents, and all matters arising out of or relating to this Agreement, whether sounding in contract, tort, or statute are governed by, and construed in accordance with, the laws of the State of [STATE], without giving effect to any conflict of law principles. [The UN Convention on Contracts for the International Sale of Goods shall not apply to this Agreement.] |
| Dispute Resolution and Jurisdiction | No ADR OR Mandatory Arbitration |
| Cumulative Remedies | N/A |
| Assignment | Favorable to Purchaser OR Neutral OR Favorable to Seller |

**Table A.2.2:** All clauses, parameters, and options from Lawgood's sale of goods contract.

| **Section** | **Parameters or options** |
|---|---|
| Entire Agreement | N/A |
| Waiver; Amendment | N/A |
| Notices | N/A |
| Severability | N/A |
| Counterparts | N/A |
| Headings | N/A |
| Force Majeure | Neutral OR Favorable to Seller |

**Table A.2.3:** All clauses, parameters, and options from Lawgood's sale of goods contract.

# B

# Arguments and evidence for DAO adoption

There are several ways that DAOs could disrupt the thousand-year-old corporate model. In order from most to least evidenced:

- Scaling funding. One demonstrated feature of DAOs is their ability to obtain funds incredibly quickly. DAOs might demonstrate a comparative advantage in raising and deploying capital given their native proximity to crypto assets such as tokens, NFTs, and various financial instruments coming from decentralized finance. DAO-based fundraising might also be especially appropriate for fast-moving software projects.

- Evading regulation. Operating on decentralized infrastructure allows (some) DAOs and blockchain operators to evade regulation within certain legal jurisdictions, e.g. U.S. securities laws or European data regulations, even when relevant persons in the DAO may reside or do business in those jurisdictions. This clearly endows DAOs with certain advantages compared to traditional entities.

- Easy to start. Registering a company requires interacting with legal authorities, possibly in multiple jurisdictions. Starting a DAO can be as easy as filling out

an online form, making them a better fit for projects that are just beginning.

- Digital jurisdiction. Blockchain-based forms of contract enforcement and alternative dispute resolution might advance to the point where companies will prefer to operate in a digital jurisdiction rather than a traditional legal jurisdiction. This is most likely to happen when the legal jurisdiction in question has low capacity or is inefficient in some way (e.g. due to corruption or lack of institutional investment), or when a company needs to operate globally across multiple jurisdictions.

- Scaling contribution. Building a product or service requires sourcing and hiring people. Open-source software has already demonstrated the ability of protocols such as git to organize contributors at large scales. DAOs, through tokens, community ownership, and other incentive mechanisms, can support scaling of contributors and mitigate some of the problems with open-source and other forms of peer production.

- Community ownership. DAOs, as variations of online communities, offer better support for large-scale community ownership and governance. In turn, community ownership might enable faster growth and better alignment between firms and their users, locking in network effects.

- Digital public goods. DAOs might be a more efficient vehicle than traditional corporations for funding, building, and protecting public goods for the entire DAO ecosystem, leading to more investment in shared infrastructure, a more interoperable institutional and technical stack, and a cheaper, more efficient ecosystem of services than that available to traditional companies.

- Consuming shared infrastructure. DAOs may not only promote relatively more investment in shared infrastructure and research but, as technical constructs, may benefit more directly and more quickly from improvements in infrastructure and research.

- Tech-based network effects. DAOs, as net-native organizations, can more efficiently exploit network effects enabled by technology and the internet.

- Investment management. Investment DAOs and DeFi DAOs might become more efficient patterns for allocating, managing, and thus growing the value of governed assets than traditional banks or private equity firms. In this view, DAOs will become holding entities for traditional corporations.

- Digital governance. Corporate governance in the form of shareholder meetings, boards of directors, and corporate charters has flaws. Digital technology already complements and automates many of these processes. DAOs represent a transition from regimes where rights are recognized and enforced through legal and formal processes to a regime where rights are recognized and enforced through code, and might provide significant efficiency gains.

- Scaling decision-making. Building and running a company requires complex decision-making at many levels, and the vast majority of corporations make these decisions through a hierarchy. DAOs are piloting decentralized forms of decision-making that, in principle, could be better suited to different business models.

- Incentives provided by states. DAOs can be mapped to both typical corporate entities such as LLCs as well as new legal entities. Those new legal entities may entail comparative legal and tax advantages for DAOs as well as other benefits for the governments and courts that adopt them.

- Long-term economic trends. The economy might rebalance into industries and arenas (e.g. the metaverse) where DAOs have a comparative advantage against traditional companies.

# References


[1] Roscoe Pound. "Law and the Science of Law in Recent Theories". en. In: *The Yale Law Journal* 43.4 (Feb. 1934), p. 525. URL: https://www.jstor.org/stable/791063?origin=crossref (visited on 10/28/2019).

[2] Roscoe Pound. "The lawyer as a social engineer". en. In: *Journal of Public Law* 3.2 (1954), p. 13.

[3] Peter Cramton. "Spectrum Auction Design". en. In: *Review of Industrial Organization* 42.2 (Mar. 2013), pp. 161–190. URL: https://doi.org/10.1007/s11151-013-9376-x (visited on 09/15/2023).

[4] Wessel Reijers et al. "Now the Code Runs Itself: On-Chain and Off-Chain Governance of Blockchain Technologies". en. In: *Topoi* (Dec. 2018). URL: https://doi.org/10.1007/s11245-018-9626-5 (visited on 04/26/2020).

[5] Avery Katz. "The Strategic Structure of Offer and Acceptance: Game Theory and the Law of Contract Formation". en. In: *Michigan Law Review* 89.2 (Nov. 1990), p. 215. URL: https://www.jstor.org/stable/1289373?origin=crossref (visited on 03/01/2022).

[6] Patrick Bolton and Mathias Dewatripont. *Contract Theory*. English. Cambridge, Mass: The MIT Press, Dec. 2004.

[7] Jay Wright Forrester. "Industrial dynamics". In: *Journal of the Operational Research Society* 48.10 (1997), pp. 1037–1041.

[8] Stafford Beer. *Brain of the firm: the managerial cybernetics of organization*. Wiley, 1972.

[9] M. W. Steenson. "5 Cedric Price: Responsive Architecture and Intelligent Buildings". In: *Architectural Intelligence: How Designers and Architects Created the Digital Landscape*. MITP, 2017, pp. 127–163. URL: https://ieeexplore.ieee.org/document/8290865.

[10] Bowen Baker et al. "Emergent Tool Use From Multi-Agent Autocurricula". In: *arXiv:1909.07528 [cs, stat]* (Feb. 2020). arXiv: 1909.07528. URL: http://arxiv.org/abs/1909.07528 (visited on 01/10/2021).

[11] Neil Ghani et al. "Compositional game theory". In: *arXiv:1603.04641 [cs]* (Feb. 2018). arXiv: 1603.04641. URL: http://arxiv.org/abs/1603.04641 (visited on 01/12/2021).

[12] Elinor Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. English. 1st edition. Cambridge University Press, Nov. 1990.

[13] Sue E. S. Crawford and Elinor Ostrom. "A Grammar of Institutions". en. In: *American Political Science Review* 89.3 (Sept. 1995), pp. 582–600. URL: https://www.cambridge.org/core/product/identifier/S0003055400097173/type/journal_article (visited on 09/20/2019).

[14] Leonid Hurwicz. "Institutions as families of game forms". en. In: *The Japanese Economic Review* 47.2 (June 1996), pp. 113–132. URL: http://doi.wiley.com/10.1111/j.1468-5876.1996.tb00038.x (visited on 08/31/2020).

[15] Lucy A Suchman. *Plans and Situated Actions: The problem of human-machine communication*. en. Tech. rep. Palo Alto Research Center: XEROC, Feb. 1985.

[16] Elinor Ostrom. *Understanding institutional diversity*. en. Princeton paperbacks. OCLC: ocm57207709. Princeton: Princeton University Press, 2005.

[17] Lawrence Lessig. *Code: Version 2.0*. Second. Basic Books, 2006.

[18] Seth Frey et al. "Composing games into complex institutions". en. In: *PLOS ONE* 18.3 (Mar. 2023). Publisher: Public Library of Science, e0283361. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0283361 (visited on 09/16/2023).

[19] Nathan Schneider et al. "Modular Politics: Toward a Governance Layer for Online Communities". In: *Proceedings of the ACM on Human-Computer Interaction* 5.CSCW1 (Apr. 2021). arXiv: 2005.13701, pp. 1–26. URL: http://arxiv.org/abs/2005.13701 (visited on 05/14/2021).

[20] Joshua Z. Tan and Luke V. Miller. "Building Net-Native Agreement Systems". en. In: *MIT Computational Law Report* (Dec. 2022). URL: https://law.mit.edu/pub/buildingnetnativeagreementsystems/release/1 (visited on 09/16/2023).

[21] Joshua Z. Tan et al. "ERC-4824: Common Interfaces for DAOs [DRAFT]". en. In: *Ethereum Improvement Proposals* 4824 (Feb. 2022). URL: https://eips.ethereum.org/EIPS/eip-4824 (visited on 10/07/2022).

[22] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. English. OCLC: 1629708. Princeton: Princeton University Press, 1944.

[23] Christian Kimmich. "Linking action situations: Coordination, conflicts, and evolution in electricity provision for irrigation in Andhra Pradesh, India". en. In: *Ecological Economics* 90 (June 2013), pp. 150–158. URL: http://www.sciencedirect.com/science/article/pii/S0921800913001080 (visited on 12/27/2020).

[24] Fritz W. Scharpf. "Games Real Actors Could Play: The Challenge of Complexity". en. In: *Journal of Theoretical Politics* 3.3 (July 1991). Publisher: SAGE Publications Ltd, pp. 277–304. URL: https://doi.org/10.1177/0951692891003003003 (visited on 12/27/2020).

[25] Jenna Bednar and Scott Page. "Can Game(s) Theory Explain Culture?: The Emergence of Cultural Behavior Within Multiple Games". en. In: *Rationality and Society* 19.1 (Feb. 2007). Publisher: SAGE Publications Ltd, pp. 65–97. URL: https://doi.org/10.1177/1043463107075108 (visited on 12/27/2020).

[26] Lee Allen Dugatkin and Hudson Kern Reeve, eds. *Game Theory and Animal Behavior*. Oxford, New York: Oxford University Press, Mar. 2000.

[27] Elinor Ostrom, Roy Gardner, and Jimmy Walker. *Rules, Games, and Common-Pool Resources*. English. Ann Arbor: University of Michigan Press, Mar. 1994.

[28] James E. Alt and Barry Eichengreen. "Parallel and Overlapping Games: Theory and an Application to the European Gas Trade*". en. In: *Economics & Politics* 1.2 (1989). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1468-0343.1989.tb00008.x, pp. 119–144. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0343.1989.tb00008.x (visited on 12/27/2020).

[29] Richard D. McKelvey, Andrew M. McLennan, and Theodore L. Turocy. "Gambit: Software tools for game theory". In: (2006). Publisher: Version 0.2006. 01.20.

[30] Edsger W. Dijkstra. "The humble programmer". en. In: *Communications of the ACM* 15.10 (Oct. 1972), pp. 859–866. URL: https://dl.acm.org/doi/10.1145/355604.361591 (visited on 02/01/2021).

[31] Donald E. Knuth. "Structured Programming with *go to* Statements". In: *ACM Computing Surveys* 6.4 (Dec. 1974), pp. 261–301. URL: https://doi.org/10.1145/356635.356640 (visited on 02/01/2021).

[32] S. Abramsky and B. Coecke. "A categorical semantics of quantum protocols". In: *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.* ISSN: 1043-6871. July 2004, pp. 415–425.

[33] John C. Baez and Blake S. Pollard. "A Compositional Framework for Reaction Networks". In: *Reviews in Mathematical Physics* 29.09 (Oct. 2017). arXiv: 1704.02051, p. 1750028. URL: http://arxiv.org/abs/1704.02051 (visited on 12/28/2020).

[34] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. "Mathematical Foundations for a Compositional Distributional Model of Meaning". In: *arXiv:1003.4394 [cs, math]* (Mar. 2010). arXiv: 1003.4394. URL: http://arxiv.org/abs/1003.4394 (visited on 11/17/2018).

[35] Jules Hedges et al. "Selection Equilibria of Higher-Order Games". en. In: *Practical Aspects of Declarative Languages*. Ed. by Yuliya Lierler and Walid Taha. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 136–151.

[36] Jules Hedges et al. "Higher-Order Decision Theory". en. In: *Algorithmic Decision Theory*. Ed. by Jörg Rothe. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 241–254.

[37] Jan Willems. "The Behavioral Approach to Open and Interconnected Systems". en. In: *IEEE Control Systems Magazine* 27.6 (Dec. 2007), pp. 46–99. URL: http://ieeexplore.ieee.org/document/4384643/ (visited on 05/18/2018).

[38] Brendan Fong. "The Algebra of Open and Interconnected Systems". In: *arXiv:1609.05382 [math]* (Sept. 2016). arXiv: 1609.05382. URL: http://arxiv.org/abs/1609.05382 (visited on 01/15/2021).

[39] Joachim Lambek. “From lambda calculus to Cartesian closed categories”. In: *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Ed. by J.P. Seldin and J. Hindley. Academic Press, 1980, pp. 376–402.

[40] Julian Hedges. “Towards compositional game theory”. PhD thesis. Queen Mary University of London, 2016.

[41] André Joyal and Ross Street. “The geometry of tensor calculus, I”. en. In: *Advances in Mathematics* 88.1 (July 1991), pp. 55–112. URL: https://www.sciencedirect.com/science/article/pii/000187089190003P (visited on 07/15/2021).

[42] John C. Baez and Mike Stay. “Physics, Topology, Logic and Computation: A Rosetta Stone”. In: *New Structures for Physics*. Ed. by Bob Coecke. Vol. 813. Lecture Notes in Physics. arXiv: 0903.0340. Berlin: Springer, 2011, pp. 95–174. URL: http://arxiv.org/abs/0903.0340 (visited on 07/15/2021).

[43] Peter Cramton et al. *Global carbon pricing: the path to climate cooperation*. The MIT Press, 2017.

[44] Peter Cramton, Axel Ockenfels, and Jean Tirole. “Translating the Collective Climate Goal Into a Common Climate Commitment”. en. In: *Review of Environmental Economics and Policy* 11.1 (Jan. 2017), pp. 165–171. URL: https://www.journals.uchicago.edu/doi/10.1093/reep/rew015 (visited on 07/15/2021).

[45] Gabriel Carroll and Ilya Segal. “Robustly optimal auctions with unknown resale opportunities”. In: *The Review of Economic Studies* 86.4 (2019). Publisher: Oxford University Press, pp. 1527–1555.

[46] Rod Garratt and Thomas Tröger. “Speculation in standard auctions with resale”. In: *Econometrica* 74.3 (2006). Publisher: Wiley Online Library, pp. 753–769.

[47] Philip A Haile. “Auctions with private uncertainty and resale opportunities”. In: *Journal of Economic theory* 108.1 (2003). Publisher: Elsevier, pp. 72–110.

[48] Paul Benjamin et al. *Institutions, Incentives, and Irrigation in Nepal*. Tech. rep. Associates in Rural Development, Inc. in collaboration with Indiana University, Workshop in Political Theory and Policy Analysis, Oct. 1994.

[49] Ganesh Shivakoti and Elinor Ostrom, eds. *Improving Irrigation Governance and Management in Nepal*. English. Oakland, CA: Ics Pr, June 2001.

[50] Philipp Zahn and Jules Hedges. *Open Game Engine*. original-date: 2019-12-05T19:43:43Z. May 2024. URL: https://github.com/CyberCat-Institute/open-game-engine (visited on 06/04/2024).

[51] Masahiko Aoki. “Linking Economic and Social-Exchange Games: From the Community Norm to CSR”. en. In: *Social Capital, Corporate Social Responsibility, Economic Behaviour and Performance*. Ed. by Lorenzo Sacconi and Giacomo Degli Antoni. London: Palgrave Macmillan UK, 2011, pp. 129–148. URL: https://doi.org/10.1057/9780230306189_6 (visited on 08/22/2020).

[52] Daron Acemoglu, Georgy Egorov, and Konstantin Sonin. "Political Selection and Persistence of Bad Governments". en. In: *The Quarterly Journal of Economics* 125.4 (Nov. 2010). Publisher: Oxford Academic, pp. 1511–1575. URL: https://academic.oup.com/qje/article/125/4/1511/1916265 (visited on 10/11/2020).

[53] Wioletta Dziuda and Antoine Loeper. "Dynamic Collective Choice with Endogenous Status Quo". In: *Journal of Political Economy* 124.4 (July 2016). Publisher: The University of Chicago Press, pp. 1148–1186. URL: https://www.journals.uchicago.edu/doi/full/10.1086/686747 (visited on 08/29/2020).

[54] Elinor Ostrom. "MULTIORGANIZATIONAL ARRANGEMENTS AND COORDINATION: AN APPLICATION OF INSTITUTIONAL ANALYSIS". en. In: *Guidance, Control, and Evaluation in the Public Sector*. Ed. by In F X Kaufmann, G Majone, and V Ostrom. Berlin and New York: Walter de Gruyter, 1986.

[55] Richard D. McKelvey, Andrew M. McLennan, and Theodore L. Turocy. *Gambit: Software Tools for Game Theory*. 2016. URL: http://www.gambit-project.org.

[56] David Mellinkoff. *The Language of the Law*. English. Eugene, Oregon: Wipf & Stock Publishers, May 2004.

[57] Louis Kaplow and Steven Shavell. *Economic Analysis of Law*. en. SSRN Scholarly Paper ID 150860. Rochester, NY: Social Science Research Network, Feb. 1999. URL: https://papers.ssrn.com/abstract=150860 (visited on 03/01/2022).

[58] Henry E. Smith. "Modularity in Contracts: Boilerplate and Information Flow". en. In: *Michigan Law Review* 104.5 (2006), pp. 163–175.

[59] Margaret Jane Radin. "Boilerplate Today: The rise of modularity and the waning of consent". In: *Michigan Law Review* 104.5 (Mar. 2006), pp. 1223–1234.

[60] Michael Genesereth. *The Legacy of Hammurabi*. en. Mar. 2021. URL: https://law.stanford.edu/2021/03/17/the-legacy-of-hammurabi/ (visited on 06/09/2024).

[61] Simon Peyton Jones, Jean-Marc Eber, and Julian Seward. "Composing Contracts: An Adventure in Financial Engineering". In: *ACM SIG-PLAN Notices*. Vol. 35. 2000.

[62] H. L. A. Hart. *The Concept of Law*. English. Ed. by Penelope Bulloch and Joseph Raz. 2nd edition. Oxford u.a: Oxford University Press, June 1997.

[63] Nick Szabo. "Smart contracts: building blocks for digital markets". In: *EXTROPY: The Journal of Transhumanist Thought* 18.2 (1996).

[64] *Lawgood*. 2022. URL: https://lawgood.io/ (visited on 10/03/2022).

[65] *YC Safe Financing Documents*. en. URL: https://www.ycombinator.com/documents (visited on 09/11/2023).

[66] John F. Coyle and Joseph M. Green. "The SAFE, the KISS, and the Note: A Survey of Startup Seed Financing Contracts Essay". eng. In: *Minnesota Law Review Headnotes* 103.1 (2018), pp. 42–66. URL: https://heinonline.org/HOL/P?h=hein.journals/headnotpan103&i=43 (visited on 09/11/2023).

[67] Ron Van der Meyden and Michael J. Maher. "Architecture for Smart SAFE Contracts". In: *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. Sept. 2021, pp. 145–148.

[68] Sridhar Nerur and VenuGopal Balijepally. "Theoretical reflections on agile development methodologies". en. In: *Communications of the ACM* 50.3 (Mar. 2007), pp. 79–83. URL: https://dl.acm.org/doi/10.1145/1226736.1226739 (visited on 06/09/2024).

[69] Eric Martínez, Francis Mollica, and Edward Gibson. "Poor writing, not specialized concepts, drives processing difficulty in legal language". en. In: *Cognition* 224 (July 2022), p. 105070. URL: https://www.sciencedirect.com/science/article/pii/S0010027722000580 (visited on 10/06/2022).

[70] Zev J. Eigen. *Experimental Evidence of the Relationship between Reading the Fine Print and Performance of Form-Contract Terms*. en. SSRN Scholarly Paper. Rochester, NY, Aug. 2011. URL: https://papers.ssrn.com/abstract=1905782 (visited on 10/06/2022).

[71] Nathan Schneider. "Admins, Mods, and Benevolent Dictators for Life: The Implicit Feudalism of Online Communities". In: *New Media & Society* (2021). URL: https://ntnsndr.in/ImplicitFeudalism (visited on 10/30/2019).

[72] Benjamin Mako Hill and Aaron Shaw. "The Wikipedia Gender Gap Revisited: Characterizing Survey Response Bias with Propensity Score Estimation". In: *PLoS ONE* 8.6 (June 26, 2013). pmid: 23840366. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3694126/ (visited on 07/09/2019).

[73] Seth Frey and Robert W. Sumner. "Emergence of Integrated Institutions in a Large Population of Self-Governing Communities". In: *PLOS ONE* 14.7 (July 11, 2019), e0216335. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0216335 (visited on 07/25/2019).

[74] Peter Ludlow, ed. *Crypto Anarchy, Cyberstates, and Pirate Utopias*. Digital Communication. Cambridge, Mass: MIT Press, 2001. 485 pp.

[75] Peter Suber. "Appendix 3: Nomic: A Game of Self-Amendment". In: *The Paradox of Self-Amendment: A Study of Law, Logic, Omnipotence, and Change*. Peter Lang, 1990. URL: https://dash.harvard.edu/handle/1/10288408 (visited on 11/06/2019).

[76] Orlando Castillo. "VOTEMGR: A Vote Manager for FidoNet Systems". In: *FidoNews* 8.35 (Sept. 2, 1991). URL: http://textfiles.com/bbs/FIDONET/FIDONEWS/fido0835.nws (visited on 10/02/2019).

[77] Christoph Carl Kling et al. "Voting Behaviour and Power in Online Democracy: A Study of LiquidFeedback in Germany's Pirate Party". In: *CoRR* abs/1503.07723 (2015). arXiv: 1503.07723. URL: http://arxiv.org/abs/1503.07723.

[78] Sam K. Jackson and Kathleen M. Kuehn. “Open Source, Social Activism and "Necessary Trade-Offs" in the Digital Enclosure: A Case Study of Platform Co-Operative, Loomio.Org”. In: *tripleC: Communication, Capitalism & Critique. Open Access Journal for a Global Sustainable Information Society* 14.2 (Oct. 14, 2016), pp. 413–427–413–427. URL: https://www.triple-c.at/index.php/tripleC/article/view/764 (visited on 10/02/2019).

[79] Eric Alston. “Constitutions and Blockchains: Competitive Governance of Fundamental Rule Sets”. In: (Mar. 21, 2019). URL: https://www.growthopportunity.org/research/working-papers/constitutions-and-blockchains (visited on 03/22/2019).

[80] Wessel Reijers et al. “Now the Code Runs Itself: On-Chain and Off-Chain Governance of Blockchain Technologies”. In: *Topoi* (Dec. 17, 2018). URL: https://doi.org/10.1007/s11245-018-9626-5 (visited on 12/19/2018).

[81] Primavera De Filippi and Aaron Wright. *Blockchain and the Law: The Rule of Code*. Harvard University Press, Apr. 9, 2018. 209 pp.

[82] Peter Kollock and Marc Smith. “Managing the Virtual Commons: Cooperation and Conflict in Computer Communities”. In: *Computer-Mediated Communication: Linguistic, Social, and Cross-Cultural Perspectives*. Ed. by Susan C. Herring. Vol. 39. Amsterdam: John Benjamins Publishing Company, 1996, p. 109. URL: https://benjamins.com/catalog/pbns.39.10kol (visited on 05/04/2019).

[83] Avi Hyman. “Twenty years of ListServ as an academic tool”. In: *The Internet and higher education* 6.1 (2003), pp. 17–24.

[84] Laurie Cubbison. “Configuring listserv, configuring discourse”. In: *Computers and Composition* 16.3 (1999), pp. 371–381.

[85] Elizabeth M Reid. *Electropolis: Communication and community on internet relay chat*. University of Melbourne, Department of History, 1991.

[86] Allucquere Rosanne Stone. “Will the real body please stand up?” In: *Cyberspace: first steps* (1991), pp. 81–118.

[87] Andrea Forte, Vanesa Larco, and Amy Bruckman. “Decentralization in Wikipedia governance”. In: *Journal of Management Information Systems* 26.1 (2009), pp. 49–72.

[88] Charles M Schweik and Robert C English. *Internet success: a study of open-source software commons*. MIT Press, 2012.

[89] Yubo Kou and Bonnie Nardi. “Regulating anti-social behavior on the Internet: The example of League of Legends”. In: *Proceedings of iConference*. 2013, 616—622.

[90] Kate Klonick. “The new governors: The people, rules, and processes governing online speech”. In: *Harv. L. Rev.* 131 (2017), p. 1598.

[91] Julian Dibbell. “A Rape in Cyberspace”. In: *The Village Voice* (Dec. 23, 1993). URL: http://www.juliandibbell.com/texts/bungle_vv.html (visited on 10/30/2019).

[92] Sanna Talja, Reijo Savolainen, and Hanni Maula. “Field differences in the use and perceived usefulness of scholarly mailing lists.” In: *Information Research: an international electronic journal* 10.1 (2004), n1.

[93] Maeve Duggan. *Online Harassment 2017*. 2017. URL: http://www.pewinternet.org/2017/07/11/online-harassment-2017/.

[94] Amanda Lenhart et al. *Online Harassment, Digital Abuse, and Cyberstalking in America*. 2017. URL: https://datasociety.net/blog/2017/01/18/online-harassment-digital-abuse.

[95] Anti-Defamation League. *» Free to Play*. 2019. URL: https://www.adl.org/free-to-play.

[96] Yochai Benkler. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. OCLC: ocm61881089. New Haven, CT: Yale University Press, 2006. 515 pp.

[97] Yochai Benkler. “Coase’s Penguin, or, Linux and "The Nature of the Firm" on JSTOR”. In: *Yale Law Journal* 112.3 (Dec. 2002), pp. 369–446. URL: https://www.yalelawjournal.org/article/coases-penguin-or-linux-and-the-nature-of-the-firm (visited on 03/03/2020).

[98] R Stuart Geiger and David Ribes. “The work of sustaining order in wikipedia: the banning of a vandal”. In: *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. 2010, pp. 117–126.

[99] Thamar Solorio, Ragib Hasan, and Mainul Mizan. “A case study of sockpuppet detection in wikipedia”. In: *Proceedings of the Workshop on Language Analysis in Social Media*. 2013, pp. 59–68.

[100] Igor Steinmacher et al. “Social barriers faced by newcomers placing their first contribution in open source software projects”. In: *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*. 2015, pp. 1379–1392.

[101] Aniket Kittur et al. “He says, she says: conflict and coordination in Wikipedia”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2007, pp. 453–462.

[102] Jane Im et al. “Deliberation and Resolution on Wikipedia: A Case Study of Requests for Comments”. In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW (2018), pp. 1–24.

[103] Nicolas Ducheneaut et al. “The life and death of online gaming communities: a look at guilds in world of warcraft”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2007, pp. 839–848.

[104] Aaron Halfaker et al. “The Rise and Decline of an Open Collaboration System: How Wikipedia’s Reaction to Popularity Is Causing Its Decline”. In: *American Behavioral Scientist* 57.5 (May 1, 2013), pp. 664–688. URL: https://doi.org/10.1177/0002764212469365 (visited on 05/26/2019).

[105] Bryan Dosono and Bryan Semaan. “Moderation practices as emotional labor in sustaining online communities: The case of AAPI identity work on Reddit”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–13.

[106] Google Jigsaw. *Perspective API*. 2017. URL: https://www.perspectiveapi.com/.

[107] Reuben Binns et al. "Like trainer, like bot? Inheritance of bias in algorithmic content moderation". In: *Social Informatics: 9th International Conference, SocInfo 2017, Oxford, UK, September 13-15, 2017, Proceedings, Part II*. Springer. Springer International Publishing, 2017, pp. 405–415. URL: https://doi.org/10.1007/978-3-319-67256-4_32.

[108] Hossein Hosseini et al. "Deceiving Google's Perspective API built for detecting toxic comments". In: *arXiv preprint arXiv:1702.08138* (2017).

[109] Charles Kiene, Andrés Monroy-Hernández, and Benjamin Mako Hill. "Surviving an" Eternal September" How an Online Community Managed a Surge of Newcomers". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 1152–1156.

[110] Brian Butler, Elisabeth Joyce, and Jacqueline Pike. "Don't look now, but we've created a bureaucracy: the nature and roles of policies and rules in wikipedia". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2008, pp. 1101–1110.

[111] Robert E Kraut et al. *The challenges of dealing with newcomers*. 2012.

[112] Katherine Panciera, Aaron Halfaker, and Loren Terveen. "Wikipedians are born, not made: a study of power editors on Wikipedia". In: *Proceedings of the ACM 2009 international conference on Supporting group work*. 2009, pp. 51–60.

[113] Aaron Shaw and Benjamin M Hill. "Laboratories of oligarchy? How the iron law extends to peer production". In: *Journal of Communication* 64.2 (2014), pp. 215–238.

[114] Oliver L Haimson and Anna Lauren Hoffmann. "Constructing and enforcing" authentic" identity online: Facebook, real names, and non-normative identities". In: *First Monday* 21.6 (2016).

[115] Tarleton Gillespie. *Custodians of the internet : platforms, content moderation, and the hidden decisions that shape social media*. eng. New Haven: Yale University Press, 2018.

[116] Electronic Frontier Foundation. *The Santa Clara Principles On Transparency and Accountability in Content Moderation*. 2018. URL: https://www.santaclaraprinciples.org.

[117] R Stuart Geiger. "Bot-based collective blocklists in Twitter: the counterpublic moderation of harassment in a networked public space". In: *Information, Communication & Society* 19.6 (2016), pp. 787–803. URL: https://ssrn.com/abstract=2761503.

[118] Sarah T Roberts. *Behind the screen: Content moderation in the shadows of social media*. Yale University Press, 2019.

[119] Kate Crawford and Tarleton Gillespie. "What is a flag for? Social media reporting tools and the vocabulary of complaint". In: *New Media & Society* 18.3 (2016), pp. 410–428.

[120] Tim Wu. *The Attention Merchants: The Epic Scramble to Get Inside Our Heads*. Knopf Doubleday Publishing Group, Sept. 5, 2017. 434 pp.

[121] Joseph Bonneau et al. *Democracy Theatre: Comments on Facebook's Proposed Governance Scheme*. Mar. 29, 2009. URL: http://www.preibusch.de/publications/Bonneau_Preibusch_Anderson_Clayton_Anderson__Facebook_Governance_Comments.pdf.

[122] Taina Bucher. "The Algorithmic Imaginary: Exploring the Ordinary Affects of Facebook Algorithms". In: *Information, Communication & Society* 20.1 (Jan. 2, 2017), pp. 30–44. URL: https://www.tandfonline.com/doi/full/10.1080/1369118X.2016.1154086 (visited on 07/15/2019).

[123] Mary L Gray and Siddharth Suri. *Ghost Work: How to Stop Silicon Valley from Building a New Global Underclass*. Eamon Dolan Books, 2019.

[124] Whitney Phillips. *This is why we can't have nice things: Mapping the relationship between online trolling and mainstream culture*. Mit Press, 2015.

[125] Cliff Lampe and Paul Resnick. "Slash (dot) and burn: distributed moderation in a large online conversation space". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2004, pp. 543–550.

[126] J Nathan Matias. "The civic labor of online moderators". In: *Internet Politics and Policy conference. Oxford, United Kingdom*. 2016.

[127] Serge Egelman. "My profile is my password, verify me! The privacy/convenience tradeoff of Facebook Connect". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2013, pp. 2369–2378.

[128] M Nouwens et al. "Dark Patterns Post-GDPR: Scraping Consent Interface Designs and Demonstrating their Influence". In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2020)*. ACM. 2020.

[129] Robert E. Kraut, Paul Resnick, and Sara Kiesler. *Building Successful Online Communities: Evidence-Based Social Design*. Cambridge, Mass: MIT Press, 2011. 1 p.

[130] Majid Yar. "Virtual Utopias and the Imaginary of the Internet". In: *The Cultural Imaginary of the Internet: Virtual Utopias and Dystopias*. Springer, 2014, pp. 27–46.

[131] Jennifer L. Mnookin. "Virtual(ly) Law: the Emergence of Law in LambdaMoo: Mnookin". In: *Journal of Computer-Mediated Communication* 2.1 (June 1996). JCMC214. URL: https://doi.org/10.1111/j.1083-6101.1996.tb00185.x.

[132] Anna DuVal Smith. "Problems of conflict management in virtual communities". In: *Communities in cyberspace*. Routledge, 2002, pp. 145–174.

[133] Brian C. Keegan and Jed R. Brubaker. *A Code Written in Sand: The Structure and Dynamics of Peer Producing Wikipedia's Rules*. New York, New York, USA, 2015.

[134] Bradi Heaberlin and Simon DeDeo. "The evolution of Wikipedia's norm network". In: *Future Internet* 8.2 (2016), p. 14.

[135] Ted Ullyot. *Results of the Inaugural Facebook Site Governance Vote*. 2019. URL: https://knightcolumbia.org/content/protocols-not-platforms-a-technological-approach-to-free-speech.

[136] Shagun Jhaver et al. “Human-machine collaboration for content regulation: The case of Reddit Automoderator”. In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 26.5 (2019), pp. 1–35.

[137] J Matias et al. “Reporting, reviewing, and responding to harassment on Twitter”. In: *Available at SSRN 2602018* (2015).

[138] Jenny Fan and Amy X Zhang. “Digital Juries: A Civics-Oriented Approach to Platform Governance”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020.

[139] Bryn Loban. “Between Rhizomes and Trees: P2P Information Systems”. In: *First Monday* 9.10 (Oct. 4, 2004). URL: `http://www.firstmonday.dk/ojs/index.php/fm/article/view/1182` (visited on 06/09/2018).

[140] Colin Harris. “Institutional Solutions to Free-Riding in Peer-to-Peer Networks: A Case Study of Online Pirate Communities”. In: *Journal of Institutional Economics* (2018), pp. 1–24.

[141] Robert Augustus Hardy and Julia R. Norgaard. “Reputation in the Internet Black Market: An Empirical and Theoretical Analysis of the Deep Web”. In: *Journal of Institutional Economics* 12.3 (2016), pp. 515–539.

[142] Alex Pazaitis, Primavera De Filippi, and Vasilis Kostakis. “Blockchain and Value Systems in the Sharing Economy: The Illustrative Case of Backfeed”. In: *Technological Forecasting and Social Change* 125 (Dec. 1, 2017), pp. 105–115. URL: `http://www.sciencedirect.com/science/article/pii/S0040162517307084` (visited on 12/04/2018).

[143] Charlotte Hess. “The Virtual CPR: The Internet as a Local and Global Common Pool Resource”. In: *Reinventing the Commons, the Fifth Biennial Conference of the International Association for the Study of Common Property*. 1995. URL: `https://dlc.dlib.indiana.edu/dlc/handle/10535/234`.

[144] Albert O. Hirschman. *Exit, Voice, and Loyalty: Responses to Decline in Firms, Organizations, and States*. Cambridge, MA: Harvard University Press, 1970. 180 pp.

[145] J. Nathan Matias. “Going Dark: Social Factors in Collective Action Against Platform Operators in the Reddit Blackout”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI ’16*. The 2016 CHI Conference. Santa Clara, CA: ACM Press, 2016, pp. 1138–1151. URL: `http://dl.acm.org/citation.cfm?doid=2858036.2858391` (visited on 07/11/2019).

[146] Alissa Centivany and Bobby Glushko. “" Popcorn Tastes Good" Participatory Policymaking and Reddit’s”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 1126–1137.

[147] Shagun Jhaver et al. “Human-Machine Collaboration for Content Regulation: The Case of Reddit Automoderator”. In: *ACM Transactions on Computer-Human Interaction* 26.5 (July 19, 2019), pp. 1–35. URL: `http://dl.acm.org/citation.cfm?doid=3349608.3338243` (visited on 11/08/2019).

[148] Eshwar Chandrasekharan et al. “Crossmod: A Cross-Community Learning-based System to Assist Reddit Moderators”. In: *Proceedings of the ACM on Human-Computer Interaction* 3.CSCW (2019), pp. 1–30.

[149] Casey Fiesler et al. “Reddit rules! characterizing an ecosystem of governance”. In: *Twelfth International AAAI Conference on Web and Social Media*. 2018.

[150] J Nathan Matias and Merry Mou. “CivilServant: Community-led experiments in platform governance”. In: *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018, pp. 1–13.

[151] J Nathan Matias. “Preventing harassment and increasing group participation through social norms in 2,190 online science discussions”. In: *Proceedings of the National Academy of Sciences* 116.20 (2019), pp. 9785–9789.

[152] Kaitlin Mahar, Amy X Zhang, and David Karger. “Squadbox: A tool to combat email harassment using friendsourced moderation”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–13.

[153] Sarita Schoenebeck, Oliver L Haimson, and Lisa Nakamura. “Drawing from justice theories to support targets of online harassment”. In: *new media & society* (2020).

[154] Lindsay Blackwell et al. “Classification and its consequences for online harassment: Design insights from heartmob”. In: *Proceedings of the ACM on Human-Computer Interaction* 1.CSCW (2017), pp. 1–19.

[155] Edward Castronova. “On the Research Value of Large Games: Natural Experiments in Norrath and Camelot”. English. In: *Games and Culture* 1.2 (Apr. 2006), pp. 163–186.

[156] Robert Scott Cavender. “Internal Reinforcement of Cooperative Outcomes: Evidence from Virtual Worlds”. English. In: *SSRN Electronic Journal* (2013).

[157] Travis L Ross and Lauren B Collister. “A Social Scientific Framework for Social Systems in Online Video Games: Building a Better Looking for Raid Loot System in World of Warcraft”. In: *Computers in Human Behavior* 36 (Jan. 2014), pp. 1–12.

[158] Pontus Strimling and Seth Frey. “Emergent Cultural Differences in Online Communities & Norms of Fairness”. In: *Working paper* (2018).

[159] Afroz Sadia et al. “Honor among Thieves: A Common’s Analysis of Cybercrime Economies”. In: *IEEE eCRS* (2013).

[160] Dan Cosley et al. “How Oversight Improves Member-Maintained Communities”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’05. Portland, Oregon, USA: Association for Computing Machinery, Apr. 2005, pp. 11–20.

[161] Tarleton Gillespie. “The Politics of ‘Platforms’”. In: *New Media & Society* 12.3 (May 1, 2010), pp. 347–364. URL: https://doi.org/10.1177/1461444809342738 (visited on 08/05/2019).

[162] James Grimmelmann. “The Virtues of Moderation”. In: 17 (2015), pp. 42–109.

[163] Joseph Seering et al. “Moderator Engagement and Community Development in the Age of Algorithms”. In: *New Media & Society* (Jan. 11, 2019). URL: https://doi.org/10.1177/1461444818821316 (visited on 05/09/2019).

[164] Jie Cai and Donghee Yvette Wohn. "What Are Effective Strategies of Handling Harassment on Twitch? Users' Perspectives". In: *Conference Companion Publication of the 2019 on Computer Supported Cooperative Work and Social Computing*. CSCW '19. Austin, TX, USA: Association for Computing Machinery, Nov. 2019, pp. 166–170.

[165] Charles Kiene, Aaron Shaw, and Benjamin Mako Hill. "Managing Organizational Culture in Online Group Mergers". In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW (Nov. 2018), 89:1–89:21.

[166] J. Blondel. *Comparative Government: An Introduction*. New York: Routledge, Jan. 27, 2014. 452 pp.

[167] Staffan I Lindberg et al. "V-Dem: A new way to measure democracy". In: *Journal of Democracy* 25.3 (2014), pp. 159–169.

[168] Zachary Elkins et al. "Constitute: The world's constitutions to read, search, and compare". In: *Journal of web semantics* 27 (2014), pp. 10–18.

[169] Susan E. Scarrow, Paul D. Webb, and Thomas Poguntke. *Organizing political parties: Representation, participation, and power*. Oxford University Press, 2017.

[170] Kenneth Janda. *Political parties: A cross-national survey*. New York: Free Press; London: Collier Macmillan, 1980.

[171] Arend Lijphart. *Patterns of Democracy: Government Forms and Performance in Thirty-six Countries*. en. Google-Books-ID: GLtX2zJrflAC. Yale University Press, 1999.

[172] Matthew Søberg Shugart. "Comparative Electoral Systems Research: The Maturation of a Field and New Challenges Ahead". en. In: *The Politics of Electoral Systems*. Ed. by Michael Gallagher and Paul Mitchell. Oxford University Press, Sept. 2005, pp. 25–56. URL: https://oxford.universitypressscholarship.com/view/10.1093/0199257566.001.0001/acprof-9780199257560-chapter-2 (visited on 09/23/2020).

[173] Stefan Voigt. "Positive constitutional economics II—a survey of recent developments". en. In: *Public Choice* 146.1 (Jan. 2011). Company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 1 Publisher: Springer US, pp. 205–256. URL: https://link.springer.com/article/10.1007/s11127-010-9638-1 (visited on 09/23/2020).

[174] Todd Landman. *Issues and Methods in Comparative Politics*. Routledge, 2000.

[175] Gert-Jan de Vreede and Gary Dickson. "Using GSS to Design Organizational Processes and Information Systems: An Action Research Study on Collaborative Business Engineering". en. In: *Group Decision and Negotiation* 9.2 (Mar. 2000), pp. 161–183.

[176] Kent Beck et al. "Manifesto for agile software development". In: (2001).

[177] Thomas W Malone et al. "Tools for Inventing Organizations: Toward a Handbook of Organizational Processes". en. In: *Management Science* 45.3 (1999), p. 20.

[178] Tilman Börgers. *An Introduction to the Theory of Mechanism Design*. Oxford University Press, July 1, 2015. URL: https://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780199734023.001.0001/acprof-9780199734023 (visited on 10/30/2019).

[179] Steve Phelps, Peter McBurney, and Simon Parsons. "Evolutionary Mechanism Design: A Review". In: *Autonomous Agents and Multi-Agent Systems* 21.2 (Sept. 1, 2010), pp. 237–264. URL: https://doi.org/10.1007/s10458-009-9108-7 (visited on 10/27/2019).

[180] Matthew O Jackson. "Mechanism theory". In: *Available at SSRN 2542983* (2014).

[181] Steven P Lalley and E Glen Weyl. "Quadratic voting: How mechanism design can radicalize democracy". In: *AEA Papers and Proceedings*. Vol. 108. 2018, pp. 33–37.

[182] Finn Kensing and Jeanette Blomberg. "Participatory design: Issues and concerns". In: *Computer supported cooperative work (CSCW)* 7.3-4 (1998), pp. 167–185.

[183] Peter M Asaro. "Transforming society by transforming technology: the science and politics of participatory design". In: *Accounting, Management and Information Technologies* 10.4 (2000), pp. 257–290.

[184] Liam Bannon, Kjeld Schmidt, and Ina Wagner. "Lest we forget". In: *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*. Springer. 2011, pp. 213–232.

[185] Elinor Ostrom. *The Institutional Analysis and Design Approach*. 1998.

[186] Elinor Ostrom. *Understanding Institutional Diversity*. Princeton, NJ: Princeton University Press, 2006. URL: https://press.princeton.edu/books/paperback/9780691122380/understanding-institutional-diversity (visited on 10/26/2019).

[187] Elinor Ostrom. "Designing Complexity to Govern Complexity". In: *Property Rights and the Environment: Social and Ecological Issues*. Ed. by Susan Hanna and Mohan Munasinghe. Washington, DC: Beijer International Institute of Ecological Economics and the World Bank, 1995.

[188] Michael D. McGinnis. "Networks of Adjacent Action Situations in Polycentric Governance". In: *Policy Studies Journal* 39.1 (2011), pp. 51–78. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1541-0072.2010.00396.x (visited on 10/27/2019).

[189] Lee Fennell. "Ostrom's Law: Property rights in the commons". In: *International Journal of the Commons* 5.1 (2011).

[190] Vincent Ostrom. "Artisanship and artifact". In: *Public Administration Review* 40.4 (1980), pp. 309–317.

[191] K. K. Shrestha and H. R. Ojha. "Theoretical Advances in Community-Based Natural Resources Management: Ostrom and Beyond". In: *Redefining Diversity Dynamics of Natural Resources Management in Asia*. Ed. by Ganesh P. Shivakoti, Ujjwal Pradhan, and Helmi. Vol. 1. Elsevier, 2017, 13–40. URL: http://www.sciencedirect.com/science/article/pii/B9780128054543000025.

[192] Peter P Mollinga. “Water and politics: levels, rational choice and South Indian canal irrigation”. In: *Futures* 33.8 (2001), 733–752. URL: http://www.sciencedirect.com/science/article/pii/S0016328701000167.

[193] Michael D. McGinnis. “An Introduction to IAD and the Language of the Ostrom Workshop: A Simple Guide to a Complex Framework”. In: *Policy Studies Journal* 39.1 (2011), pp. 169–183. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1541-0072.2010.00401.x (visited on 09/24/2019).

[194] Elinor Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge, UK: Cambridge University Press, Nov. 30, 1990. 308 pp.

[195] Seth Frey, P. M. Krafft, and Brian C. Keegan. “"This Place Does What It Was Built For": Designing Digital Institutions for Participatory Change”. In: *Proc. ACM Hum.-Comput. Interact.* 3 (CSCW Nov. 2019), 32:1–32:31. URL: http://doi.acm.org/10.1145/3359134 (visited on 11/12/2019).

[196] Vincent Ostrom. “Polycentricity (part 1)”. In: *Polycentricity and local public economies: Readings from the workshop in political theory and policy analysis*. Ann Arbor: University of Michigan Press. 1999, pp. 52–74.

[197] Elinor Ostrom. “Beyond markets and states: polycentric governance of complex economic systems”. In: *American economic review* 100.3 (2010), pp. 641–72.

[198] Fernanda B. Viegas, Martin Wattenberg, and Matthew M. McKeon. “The Hidden Order of Wikipedia”. In: ed. by D. Schuler. Berlin: Springer Verlag, 2007, pp. 445–454.

[199] M. Six Silberman. “Reading Elinor Ostrom In Silicon Valley: Exploring Institutional Diversity on the Internet”. In: *Proceedings of the 19th International Conference on Supporting Group Work*. GROUP ’16. New York: ACM, 2016, pp. 363–368. URL: http://doi.acm.org/10.1145/2957276.2957311 (visited on 07/10/2018).

[200] Niels Seidel. “Democratic Power Structures in Virtual Communities”. In: *Proceedings of the 24th European Conference on Pattern Languages of Programs*. EuroPLop ’19. Irsee, Germany: ACM, 2019, 31:1–31:8. URL: http://doi.acm.org/10.1145/3361149.3361181 (visited on 12/19/2019).

[201] Diego Giannone. “Political and Ideological Aspects in the Measurement of Democracy: The Freedom House Case”. In: *Democratization* 17.1 (Feb. 1, 2010), pp. 68–97. URL: https://doi.org/10.1080/13510340903453716 (visited on 09/28/2019).

[202] Larry Diamond and Leonardo Morlino. “The Quality of Democracy: An Overview”. In: *Journal of Democracy* 15.4 (Oct. 13, 2004), pp. 20–31. URL: https://muse.jhu.edu/article/173999 (visited on 10/18/2019).

[203] Margaret M. Burnett. “Visual Programming”. In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. American Cancer Society, 1999. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W1707 (visited on 12/26/2019).

[204] Stephen Wolfram. *What We've Built Is a Computational Language (and That's Very Important!)* May 9, 2019. URL: https://writings.stephenwolfram.com/2019/05/what-weve-built-is-a-computational-language-and-thats-very-important/ (visited on 12/26/2019).

[205] Mike Masnick. *Protocols, Not Platforms: A Technological Approach to Free Speech*. 2019. URL: https://knightcolumbia.org/content/protocols-not-platforms-a-technological-approach-to-free-speech.

[206] Jo Freeman. "The Tyranny of Structurelessness". In: *Berkeley Journal of Sociology* 17 (1972), pp. 151–164. URL: https://www.jstor.org/stable/41035187 (visited on 08/07/2019).

[207] Noam Cohen. "A 1970s Essay Predicted Silicon Valley's High-Minded Tyranny". In: (Nov. 15, 2018). URL: https://www.wired.com/story/silicon-valley-tyranny-of-structurelessness/ (visited on 03/03/2020).

[208] Langdon Winner. "Do artifacts have politics?" In: *Readings in the Philosophy of Technology* (2004), pp. 251–263.

[209] Stefania Milan and Emiliano Treré. "Big Data from the South(s): Beyond Data Universalism:" en. In: *Television & New Media* (Apr. 2019).

[210] Trebor Scholz and Nathan Schneider. *Ours to Hack and to Own: The Rise of Platform Cooperativism, a New Vision for the Future of Work and a Fairer Internet*. OCLC: 957955797. New York: OR Books, 2016.

[211] David Bollier and Silke Helfrich. *Patterns of Commoning*. Commons Strategy Group and Off the Common Press, Nov. 6, 2015. 476 pp.

[212] Daniel DeCaro. "Humanistic Rational Choice and Compliance in Motivation in Complex Societal Dilemmas". In: *Contextualizing Compliance in the Public Sector: Individual Motivations, Social Processes, and Institutional Design*. Ed. by Saba Siddiki, Salvador Espinosa, and Tanya Heikkila. New York: Routledge, July 10, 2018, p. 126. URL: https://doi.org/10.4324/9781315148144.

[213] Legal Information Institute. *Agreement*. en. Dec. 2021. URL: https://www.law.cornell.edu/wex/agreement (visited on 12/28/2021).

[214] eBay Inc. *eBay unpaid item policy*. en. Dec. 2021. URL: https://www.ebay.com/help/policies/payment-policies/unpaid-item-policy?id=4271 (visited on 12/28/2021).

[215] Mathias Reimann and Reinhard Zimmerman. *The Oxford Handbook of Comparative Law*. Oxford University Press, Nov. 2006. URL: https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199296064.001.0001/oxfordhb-9780199296064 (visited on 12/29/2021).

[216] Robert Merkin and Séverine Saintier. *Poole's Textbook on Contract Law*. 15th. Oxford, New York: Oxford University Press, Aug. 2021.

[217] Daniel Markovits and Emad Atiq. "Philosophy of Contract Law". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2021. Metaphysics Research Lab, Stanford University, 2021. URL: https://plato.stanford.edu/archives/win2021/entries/contract-law/ (visited on 12/27/2021).

[218] Vitalik Buterin. *A next-generation smart contract and decentralized application platform*. en. 2013. URL: https://ethereum.org (visited on 12/28/2021).

[219] Jae Kwon and Ethan Buchman. "Cosmos Whitepaper". en. URL: https://cosmos.network (visited on 12/29/2021).

[220] Sari Azout. *Ghost Knowledge*. Accessible at https://web.archive.org/web/20210708204302/https://www.ghostknowledge.com/. URL: https://www.ghostknowledge.com/ (visited on 01/13/2022).

[221] Sari Azout. *Interview with Sari Azout*. Sept. 2021.

[222] Vitalik Buterin. *DAOs, DACs, DAs and More: An Incomplete Terminology Guide*. en. June 2014. URL: https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/ (visited on 06/30/2021).

[223] Stan Larimer. *Bitcoin and the Three Laws of Robotics*. en. Sept. 2013. URL: https://letstalkbitcoin.com/blog/post/bitcoin-and-the-three-laws-of-robotics (visited on 06/30/2021).

[224] Quinn DuPont. "Experiments in algorithmic governance: A history and ethnography of "The DAO," a failed decentralized autonomous organization". In: *Bitcoin and Beyond*. Num Pages: 21. Routledge, 2017.

[225] Samuel Haig. "The number of active DAOs is up 660% since 2019". In: *Cointelegraph* (Sept. 2020). URL: https://cointelegraph.com/news/the-number-of-active-daos-is-up-660-since-2019 (visited on 09/01/2023).

[226] Nilay Patel. *From a meme to $47 million: ConstitutionDAO, crypto, and the future of crowdfunding*. en-US. Dec. 2021. URL: https://www.theverge.com/22820563/constitution-meme-47-million-crypto-crowdfunding-blockchain-ethereum-constitution (visited on 03/28/2023).

[227] Samer Hassan and Primavera De Filippi. "Decentralized Autonomous Organization". en. In: *Internet Policy Review* 10.2 (Apr. 2021). URL: https://policyreview.info/glossary/DAO (visited on 10/07/2022).

[228] Trebor Scholz and Nathan Schneider, eds. *Ours to Hack and to Own: The Rise of Platform Cooperativism, A New Vision for the Future of Work and a Fairer Internet*. OR Books, 2016. URL: https://www.jstor.org/stable/j.ctv62hfq7 (visited on 09/17/2023).

[229] Morshed Mannan. *Fostering Worker Cooperatives with Blockchain Technology: Lessons from the Colony Project*. en. SSRN Scholarly Paper. Rochester, NY, Dec. 2018. URL: https://papers.ssrn.com/abstract=3356774 (visited on 09/17/2023).

[230] David Ronfeldt. *Tribes, Institutions, Markets, Networks: A Framework About Societal Evolution*. en. Tech. rep. RAND Corporation, Jan. 1996. URL: https://www.rand.org/pubs/papers/P7967.html (visited on 09/17/2023).

[231] Kei Kreutler. "A Prehistory of DAOs". In: *Gnosis Guild Mirror* (July 2021). URL: https://gnosisguild.mirror.xyz/t4F5rItMw4-mlpLZf5JQhElbDfQ2JRVKAzEpanyxW1Q (visited on 09/07/2023).

[232] Vili Lehdonvirta and Edward Castronova. *Virtual economies : design and analysis*. English. Information policy series. Cambridge, MA: MIT Press, 2014. URL: http://www.jstor.org/stable/10.2307/j.ctt9qf5t6 (visited on 01/11/2019).

[233] Yochai Benkler. "Peer production, the commons, and the future of the firm". en. In: *Strategic Organization* 15.2 (May 2017), pp. 264–274. URL: http://journals.sagepub.com/doi/10.1177/1476127016652606 (visited on 09/17/2023).

[234] *Decentralized Autonomous Organization Toolkit*. Tech. rep. World Economic Forum, Jan. 2023.

[235] Joshua Z. Tan. *A long bet for DAOs*. May 2022. URL: https://mirror.xyz/thelastjosh.eth/KJVBCZhszM3Gl6VkMMcxKsR47au0xtt-dlcvZ644yKg (visited on 09/18/2023).

[236] Lucia Korpas and Joshua Z. Tan. *Governance Surfaces of DAOs*. 2023. URL: http://govbase.metagov.org/ (visited on 03/31/2023).

[237] Joshua Z. Tan et al. "The Constitutions of Web3". Metagovernance Project, 2022. URL: https://constitutions.metagov.org/ (visited on 03/31/2023).

[238] Lucia M. Korpas, Seth Frey, and Joshua Tan. "Political, economic, and governance attitudes of blockchain users". In: *Frontiers in Blockchain* 6 (2023). URL: https://www.frontiersin.org/articles/10.3389/fbloc.2023.1125088 (visited on 08/27/2023).

[239] Primavera De Filippi and Aaron Wright. *Blockchain and the Law: The Rule of Code*. English. Cambridge, Massachusetts: Harvard University Press, Apr. 2018.

[240] Nathan Schneider. "How We Can Encode Human Rights In The Blockchain". en-US. In: (June 2022). URL: https://www.noemamag.com/how-we-can-encode-human-rights-in-the-blockchain (visited on 09/07/2023).

[241] Vitalik Buterin. "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform." en. Dec. 2014.

[242] Jon Agar. *The government machine: a revolutionary history of the computer*. en. History of computing. Cambridge, Mass: MIT Press, 2003.

[243] A. Newell and H. Simon. "The logic theory machine–A complex information processing system". In: *IRE Transactions on Information Theory* 2.3 (Sept. 1956). Conference Name: IRE Transactions on Information Theory, pp. 61–79.

[244] Herbert A. Simon. *Administrative Behavior, 4th Edition*. English. 4th Revised ed. edition. New York: Free Press, Mar. 1997.

[245] Marvin Minsky. *The society of mind*. en. New York: Simon and Schuster, 1986.

[246] Nils J. Nilsson. *Shakey the Robot*. Tech. rep. Technical Note 323. AI Center, SRI International, Apr. 1984. URL: http://www.ai.sri.com/pubs/files/629.pdf (visited on 01/08/2021).

[247] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. en. Prentice Hall series in artificial intelligence. Englewood Cliffs, N.J: Prentice Hall, 1995.

[248] Azeem Azhar. *Beneficial Artificial Intelligence*. URL: https://hbr.org/podcast/2019/06/beneficial-artificial-intelligence (visited on 01/11/2021).

[249] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf (visited on 11/18/2018).

[250] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *arXiv:1412.6572 [cs, stat]* (Mar. 2015). arXiv: 1412.6572. URL: http://arxiv.org/abs/1412.6572 (visited on 01/11/2021).

[251] Deepak Pathak et al. "Curiosity-driven Exploration by Self-supervised Prediction". In: *arXiv:1705.05363 [cs, stat]* (May 2017). arXiv: 1705.05363. URL: http://arxiv.org/abs/1705.05363 (visited on 01/11/2021).

[252] Sebastian Flennerhag et al. "Meta-Learning with Warped Gradient Descent". In: *arXiv:1909.00025 [cs, stat]* (Feb. 2020). arXiv: 1909.00025. URL: http://arxiv.org/abs/1909.00025 (visited on 01/11/2021).

[253] Jonathan Lorraine, Paul Vicol, and David Duvenaud. "Optimizing Millions of Hyperparameters by Implicit Differentiation". In: *arXiv:1911.02590 [cs, stat]* (Nov. 2019). arXiv: 1911.02590. URL: http://arxiv.org/abs/1911.02590 (visited on 01/11/2021).

[254] Yoav Freund and Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting." In: *Journal of Computer and System Sciences* 55 (1997).

[255] David Balduzzi. *Composition, learning, and games*. Invited talk. Vancouver, Dec. 2019.

[256] Brendan Fong, David I. Spivak, and Rémy Tuyeras. "Backprop as Functor: A compositional perspective on supervised learning". In: *arXiv:1711.10455 [cs, math]* (May 2019). arXiv: 1711.10455. URL: http://arxiv.org/abs/1711.10455 (visited on 01/15/2021).

[257] Jules Hedges. *From open learners to open games*. Tech. rep. arXiv: 1902.08666. Feb. 2019. URL: http://arxiv.org/abs/1902.08666 (visited on 01/08/2021).

[258] Margaret Hagan. *Quick takes on how to bring prototyping into policy-making*. en. Nov. 2018. URL: https://medium.com/legal-design-and-innovation/quick-takes-on-how-to-bring-prototyping-into-policy-making-cc720f306d93 (visited on 09/18/2023).