

Computational Governance and Violable Contracts for Blockchain Applications

Munindar P. Singh, North Carolina State University

Amit K. Chopra, Lancaster University

Blockchain technology has brought newfound prominence to the challenges of building decentralized systems, which we understand quite literally as systems with no distinguished locus of control. As such, blockchain is a natural fit for building systems that support interactions among autonomous parties, each an independent locus of control. Unsurprisingly, blockchain promises support for multiparty interactions in domains such as government, health care, manufacturing, and banking.¹

Blockchain applications, which were conceived to upend conventional business models, rely upon a smart contract: code placed in and executed from a blockchain (see “Blockchain and Smart Contracts, Conceptually”). But smart contracts suffer from major shortcomings that undermine their usefulness for decentralized applications. Specifically, smart contracts are antithetical to autonomy and compatible only with endogenous applications: those computed entirely within a blockchain. Thus, smart contracts are inadequate for real applications (consider health care,

We propose a sociotechnical, yet computational, approach to building decentralized applications that accommodates and exploits blockchain technology. Our architecture incorporates the notion of a declarative, violable contract and enables flexible governance based on formal organizational structures, correctness verification without obstructing autonomy, and a basis for trust.

finance, and the Internet of Things) that typically involve external components.

ARGUMENT: VIOLABILITY, VERIFIABILITY, VALIDATION

Decentralized applications presuppose modeling interactions between autonomous parties, which calls for a representation of contracts. A crucial property of any contract is verifiability: It should be possible to determine from a public record of events whether the contract was satisfied or violated. Verifiability lies at the heart of public semantics.² Whereas smart contracts seek to prevent violation, we embrace violability and make verifiability explicit.

Verifiability requires a formal representation of a contract to computationally evaluate a history of attestations. Validation, ensuring that stakeholder requirements are correctly captured, presumes a high-level language that provides relevant abstractions.

Accordingly, we formulate a perspective on sociotechnical systems whose salient features are 1) an autonomy-preserving representation for violable contracts, 2) guaranteed verifiability through formal semantics interpreted over blockchain, 3) high-level representation to facilitate validation,

and 4) an architecture of organizations that balances flexibility and rigor to engender trust.

SCOPE AND CONTRIBUTIONS

We focus on sociotechnical challenges, deemphasizing concerns such as confidentiality and performance, and contribute

- › an analysis of the shortcomings of smart contracts through the lens of decentralized applications
- › a formulation of the research challenges to address those

shortcomings from a sociotechnical perspective

- › a description of the key elements of a possible solution.

SOCIOTECHNICAL LIMITATIONS OF SMART CONTRACTS

Let's consider the hazards of smart contracts. The Decentralized Autonomous Organization (DAO) fiasco¹⁶ is telling. The DAO, a venture-funding entity created as a smart contract on the Ethereum blockchain, was hacked to the tune of US\$50 million by exploiting

BLOCKCHAIN AND SMART CONTRACTS, CONCEPTUALLY

Notionally, a blockchain is an immutable distributed ledger, as epitomized by Bitcoin.^{S1} Blockchain solves the longstanding distributed-computing problem of achieving immutable consensus on the state of the system, despite failures and malice. Here, immutability relies upon consensus, which depends on a majority of the computing power on the network remaining in the hands of benevolent (that is, protocol-following) parties. Specifically, blockchain determines a consensus order in which events have occurred.

The idea of a smart contract^{S2} predates blockchain. A smart contract specifies conditions programmatically so that it automatically executes when the input data meet the stated terms. A vending machine is characterized as a smart contract that takes in coins and outputs a product. Smart contracts could potentially be attached to any real-world object, for example, a house for rent.^{S3} In blockchain applications, a smart contract is digitally

signed by its creator and placed on a blockchain. Since a smart contract is public, the parties wishing to exercise it can know in advance how it will function, provided that they can understand the associated program. Hence, smart contracts can enable commerce in an open setting. Bitcoin transactions are simple smart contracts. Bitcoin's limited language facilitates little more than verifying signatures. But subsequent approaches, including Ethereum, ambitiously support Turing-complete languages for smart contracts that initiate transactions based on observed events.

References

- S1. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Bitcoin. Accessed on: Nov. 6, 2019. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- S2. Wikipedia, "Smart contract." Accessed on: June 18, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Smart_contract
- S3. Slock.it. Accessed on: Nov. 6, 2019. [Online]. Available: <https://slock.it>

a flaw in the smart contract and the underlying Ethereum virtual machine. The specific flaw does not concern us since it is merely a symptom of an imperfect architecture that confuses verifiability with inviolability.

Interestingly, the hack was remedied by causing a fork in the blockchain. Specifically, several Ethereum users colluded to extend a prior block to exclude the undesirable transactions, which resulted in discarding legitimate activities, as well. (Naturally, this effort produced two competing versions of Ethereum, although the details of their history don't concern us here.) Of course, a fork was possible only because a large fraction of the active participants agreed to it. A minority would not be able to take such remedies.

For something small like the DAO, it may be appropriate to discard several days of legitimate transactions to avert a loss of US\$50 million. But what would the tradeoffs be in practice? Would it be fair to discard an hour's worth of real commerce at the national scale to save US\$50 million? We suspect not. A patient attacker may succeed by causing only small amounts of harm at a time, for which detection and reversion are infeasible.

The success of the fork, however, undermines the very point that motivated blockchains, that is, their immutability. The episode reinforces the main claim of this article: There is necessarily a social underpinning to any approach that has pretensions to decentralization. On permissioned blockchains, such as Hyperledger,¹⁷ where membership is controlled, the risk is presumably better contained. However, errors in smart contracts are unavoidable, and undesirable outcomes can be difficult to reverse.

In essence, our main choice is whether to 1) keep the social component ad

hoc, hidden, and second class, as existing approaches do, or 2) make the social component principled, explicit, and computational, as we propose to do. We now discuss three major shortcomings in the current conception of smart contracts and formulate questions that guide our investigation.

Lack of control

The independence of the participants with respect to their beliefs and actions is a crucial aspect of decentralization. Blockchain supports independence with regard to private beliefs, since consensus applies only to shared events, with such agreement being essential for achieving interoperation. However, smart contracts fail in terms of independence for actions. They automate processing, which removes control from the participants. A smart contract, once launched, cannot be overridden. Indeed, we cannot even contemplate overriding a smart contract because it executes automatically. How can we reconcile blockchain with participant autonomy?

Lack of understanding

Since the meaning of a smart contract is hidden in a procedure, which may or may not be public, one cannot readily determine whether the agreement meets the stakeholders' requirements and how it may be exercised by a participant. Since blockchains are immutable, any mistake in capturing requirements cannot be corrected without violating the immutability. Therefore, a powerful language for smart contracts placed on a blockchain poses a huge risk, as the DAO incident illustrates. Instead, we need a language in which we can capture the essential stakeholder requirements directly. To enhance users' confidence that it would capture requirements correctly, such a

language would offer constructs close to the stakeholders' conception and be limited in its expressiveness. How can we develop a contract language with an appropriate semantics?

Lack of social meaning

Any software application involves contact with the real world. In rare cases, the real world can be abstracted out. Bitcoin, being designed for cryptocurrency, is endogenous—bitcoins exist entirely within the blockchain, which can, therefore, ensure their integrity. Bitcoin is an atypical blockchain application since it excludes considerations other than transactions involving bitcoins.

More commonly, applications such as health care and commerce are entwined with the real world, both social and technical. For example, in health care, surgical equipment may fail, or a patient may deny having been adequately informed when giving consent. For physical and communication failures, the possible resolutions lie in the social sphere, as traditionally handled through contracts and laws. The DAO hack demonstrated an integrity violation, indicating a platform failure. In a decentralized scenario, any response to an interoperation failure, including a platform breakdown, must be social. Indeed, the response of forking the Ethereum blockchain was social; it was an ad hoc and unverifiable response entirely outside the computational realm. How can we enhance blockchain with abstractions to express and compute social meaning?

ARCHITECTURE: COMPACTS, GOVERNANCE, VERIFICATION, AND TRUST

The aforementioned discussion shows that smart contracts are inadequate for

describing interoperation between autonomous parties. They take over control of participant decision making, are opaque, and omit social meaning. We now describe our architecture that avoids these shortcomings and enables natural interactions between autonomous parties.

Declarative violable contracts

We introduce the term “compact”¹⁹ for our conception of contracts to avoid confusion with smart and traditional contracts. In contrast to a smart contract, a compact is not a program executed by the blockchain but a specification of correct behavior. Contrary to a traditional contract, a compact is a computational artifact; its formal semantics determines which blockchain instances satisfy the compact and which violate it. A compact would be stored on the blockchain and unambiguously computed based on its semantics. Figure 1 illustrates how compacts differ from smart contracts. In both settings, principals (social

entities) own and control devices (technical entities) such as computers, sensors, and vehicles. Importantly, the autonomy rests with social entities that control the technical entities.

A device may originate an event or relay an occurrence from another source, such as a human. The blockchain records the events it receives from devices that pass any input checks. Smart contracts provide two functions in both architectures, and in both, the input checker is a smart contract. In the traditional architecture [Figure 1(a)], the principals additionally specify their business agreements as smart contracts that carry out actions and record events on the blockchain. Thus, a smart contract, once launched, may perform immutable (modulo rollback and forking, as in the DAO incident) changes to the blockchain.

In our proposed architecture [Figure 1(b)], the principals specify the compacts corresponding to their business agreements. Given the recorded events, the evaluator—which is a

smart contract by virtue of being a program on the blockchain—determines whether a compact is satisfied, violated, expired, or neither. It informs the principals about the states of relevant compacts but does not insert events into the blockchain. Of the smart-contract functions in Figure 1(a), the architecture in Figure 1(b) retains the ability to check (filter) incoming events and query the blockchain but does not make changes to the blockchain based solely on the compacts.

In general, our sociotechnical architecture splits the necessary functions between the social and technical parts (and models the social part computationally), whereas the traditional architecture hides the social part and places all functions in the technical part. Through our models of compacts, organizations, and trust, the rest of this article demonstrates how the social and technical parts coexist. For expository convenience, we place input checking as a technical function; in the limit, we could potentially dispense with input checking and handle all discrepancies at the social level, although it would not be an effective approach for many practical applications.

A compact helps balance autonomy and correctness. A party to a compact, in exercising its autonomy, may violate the compact. For example, a compact in health care may specify that a hospital prohibits a nurse from sharing a patient’s data without consent. Yet a nurse Bob may share patient Charlie’s data with cardiologist Alice without Charlie’s consent. From the semantics, given recorded events on the blockchain, we can compute whether the compact was satisfied or violated. Crucially, violation doesn’t entail malfeasance. It could be that Charlie had a medical emergency and was in no

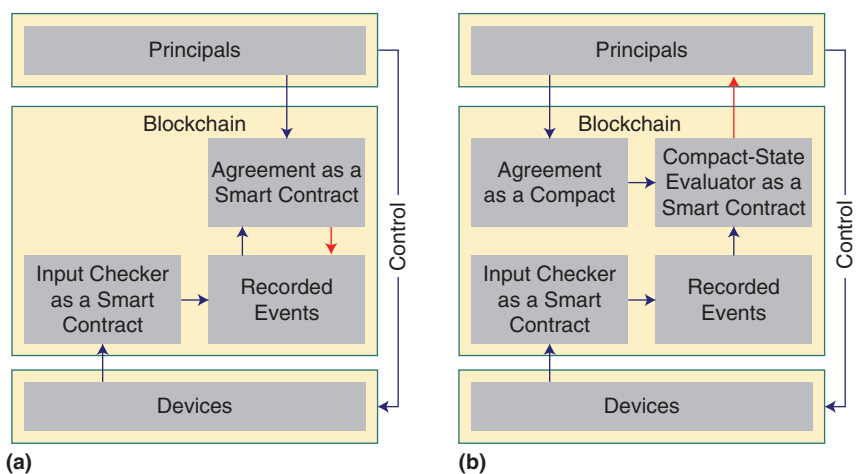


FIGURE 1. The comparison of compacts and smart contracts in a blockchain architecture. (a) Smart contracts not only check received events but can also insert additional events into the blockchain. (b) Compacts provide a declarative standard of correctness but do not insert events into the blockchain.

condition to give consent. Bob could be rewarded for saving Charlie's life.³

Consider another example: a compact for renting apartments to tourists. Such a compact may stipulate conditions, for example, that registered guests may not smoke in the apartment, invite others except children younger than 12 years old to stay overnight, and leave the windows open during the day. Such prohibitions are impossible to impose through the blockchain since they concern exogenous events and would be impractical or risky to enforce physically. Guests may violate the stipulations in the compact. The rental agency may install monitors (for instance, smoke detectors, face recognizers, and window sensors) that enable the detection of violations. Again, from the semantics and recorded events, we can compute whether the compact was satisfied or violated. As before, violation doesn't entail malfeasance. First, sensors aren't perfect, and a smoke detector may falsely report smoking, for example, due to a deep fryer. Second, a compact may be overridden by other imperatives (don't leave a child alone), or a principal may discover that circumstances make it is sensible to violate an agreement. For example, if the adult guests are taken to a hospital, they may hire a babysitter to stay overnight with their young children even though doing so is prohibited by the compact. The foregoing examples highlight the importance of compacts in detecting and resolving conflicting requirements.⁴

Specifying compacts via norms

To recover understanding and control and make the social meaning explicit, we need a declarative representation for compacts that captures

the essence of traditional contracts. A compact would explicitly state what each concerned party may expect from another. To this end, the formal notion of norms, which resembles but is not identical to "social norms" in the vernacular, yields promising constructs. As motivated by Georg von Wright, who invented modern deontic logic during the 1950s, this notion of the norm carries regulatory force.⁵ Therefore, we propose to represent each compact as a set of norms. The specific norms we adopt are commitment, authorization, prohibition, and power. The following are key features of norms:

- › Each norm in our representation is directed from its accountable party to its party with standing.⁶ Thus, a norm always makes accountability clear.
- › Each norm arises in the context of an organization. Thus, a norm makes its scope and adjudicating jurisdiction clear.
- › Each norm is conditional and states the logical conditions under which it goes into force (antecedent) and under which it concludes (consequent). The antecedent and consequent are definitively evaluated on a ledger, thereby ensuring clarity on what state each norm instance is in.

Let's introduce our specification language, based on the Custard language,⁷ via an example. We'll begin with a routine business agreement, which may be described by a compact that constitutes the following commitment. In it, keywords appear in a sans-serif font. Words beginning with an uppercase letter are names of event schemas unless otherwise specified.

Words beginning with a lowercase letter are attributes of the events.

```
compact Market
role Seller Buyer Marketplace

commitment DiscountQuote from Seller
    to Buyer within Marketplace
create Quote
detach (Order and Payment)
    deadline Quote + 10m
    where paymentAmount >=
        0.90 * quotedPrice * quantity
discharge Shipment deadline
    Payment + 5d
```

This listing describes a compact, Market, consisting of one commitment schema labeled DiscountQuote, which is directed from a role Seller to a role Buyer. At runtime, these roles are played by specific principals, for example, individuals Meryl and Custer.

- › An instance of DiscountQuote is created when an instance of the Quote event occurs, with Quote being the event expression given under create. The attributes of Quote, such as quoteID, item, and quotedPrice, are the information relevant to the creation of this commitment. Meryl alone can commit herself.
- › A (created) instance of DiscountQuote is detached when instances of Order and Payment for a matching quoteID occur within 10 min of the matching Quote, and paymentAmount is at least 90% of the cost of the items in Order (quotedPrice × quantity). Here, Custer would bring about those events, although, in general, a commitment could be detached through anyone's actions.

- › An instance of *DiscountQuote* expires if a matching *Order* and *Payment* do not occur within 10 min of the matching *Quote*.
- › If *Shipment* for the matching *quoteID* occurs (and, if there is a matching instance of *Payment*, within five days of *Payment*), the commitment is discharged. Presumably, Meryl or one of her business partners would bring about this event.
- › If the commitment is detached but *Shipment* for the matching *quoteID* does not occur within five days of *Payment*, the commitment is violated. Now, Custer can hold Meryl to account.

We illustrate the previous health-care example. For brevity, we focus on a prohibition norm and assume the relevant events: *Employment*, when a Nurse is hired by a Hospital; *DataAccess*, when a Nurse accesses a Patient’s data; and *CopyData*, when a Nurse shares a Patient’s data with someone. The attributes of these events express relevant information, including the Patient’s identity. The compact specifies a prohibition on a Nurse by a Hospital that is created when the Nurse is employed. When the Nurse accesses a Patient’s data, the Nurse may not copy that data to anyone outside of the Patient’s care team.

```
compact PatientData
role Patient Hospital Nurse
  MedicalSystem CareTeam
```

```
prohibition NoSharing on Nurse by
  Hospital within MedicalSystem
create Employment /* in Hospital */
detach DataAccess /* about Patient */
violate CopyData /* to receiver */
where receiver not in CareTeam
  of Patient /* receiver is an outsider */
```

Computing the norm lifecycle

The declarative specification of compacts yields significant benefits over smart contracts. First, the language of norms is geared toward expressing agreements between autonomous principals, and norms can be reliably identified from real-life natural-language contracts.⁸ Two, the language is amenable to formal reasoning since it is simpler than a traditional programming language. Three, specifications in the language can be automatically evaluated, meaning that the state of any norm can be unambiguously determined from the norm’s expression and the events recorded in any snapshot of the blockchain.

Specifically, a blockchain is naturally modeled as a sequence of events with timestamps. From these events, we can determine what norm instances have been created and which of them have transitioned to other relevant states. For example, we might observe that Meryl has produced instances of *Quote* for 10 prospective buyers, of whom Custer alone has responded with matching instances of *Order* and *Payment*. From the matching *Shipment* event, we can conclude that Meryl discharged her commitment to Custer.

Alternatively, if Meryl failed to bring about a matching *Shipment* event, we would conclude that Meryl violated her commitment to Custer. Importantly, we can compute abstract events, such as when a norm instance transitions in its lifecycle. For example, the violation of a commitment is itself an event that we can effectively compute. That event could be referenced from other norms, which is essential to achieving governance, as discussed next.

Organizations and governance

Consensus on what has transpired can support decentralized applications by

averting disputes about the public facts. But as envisioned here, the principals may, nevertheless, violate applicable compacts. Decentralized applications cannot avoid governance; the choice is 1) leave governance ad hoc and manual or 2) make it formal and computational, as we envision. In our conception, every decentralized application is associated with an organization, which serves as the context of its defining compact. Such organizations are seen on today’s blockchains, for instance, as channels on Hyperledger Fabric. However, current practice doesn’t model the organization itself. The ill-fated DAO was modeled procedurally as a smart contract, but that is not satisfactory since, even if it were correct, its behavior would not have been comprehensible or modifiable.

Today’s approaches lack a computational model for such organizations. Consequently, there is no precise characterization of what an organization can expect from its members and vice versa. As a result, governance in blockchain applications remains ad hoc. To address this limitation, we propose a three-pronged approach. First, we model an organization as a principal on par with any other, for instance, an individual. An organization may feature as a subject or object of another norm. Second, an organization provides an organizational context for each norm that arises, as described in Singh.⁹ In the previous listing, the Marketplace role provides the context for the Seller and Buyer’s dealings. The Marketplace role would be adopted by a concrete organization, such as the Artspllosure arts fair in Raleigh, North Carolina, within whose scope Meryl and Custer would interact. The context can embody jurisdictional weight and serve as an adjudicating authority for disputes. The

context can, thus, help mitigate violations of the norms in a compact.¹⁰

The Marketplace serves as the context for the DiscountQuote commitment and the MedicalSystem as the context for the NoSharing prohibition. The organizational context is captured as a role in its own right. Let us extend the earlier examples to illustrate how compacts can handle violations. Marketplace makes a commitment to Buyer that if Seller violates the DiscountQuote commitment, it will step in and provide a refund within two days.

```
commitment Compensation Marketplace
to Buyer within Marketplace
create Quote /* when Seller creates a
Quote */
detach violated(DiscountQuote)
discharge Refund deadline violated
(DiscountQuote) + 2d
where refundAmount =
paymentAmount
```

Similarly, MedicalSystem, as the context of the NoSharing prohibition norm, commits to Patient that it will investigate any violations of the NoSharing prohibition within 30 days.

```
commitment SanctionC from MedicalSystem
to Patient within MedicalSystem
create Enroll /* when Patient enrolls */
detach violated(NoSharing)
/* the Patient ID match is implicit */
discharge Investigation deadline
violated(NoSharing) + 30d
```

Now, when Bob reveals Charlie's data without consent, the medical system's commitment to Charlie is activated. The medical system can satisfy its commitment by conducting its investigation, upon which it may exonerate and reward Bob or penalize him.

Third, the organization is specified through a compact between itself and its members. This compact stipulates precisely what expectations an organization and its members may have of each other. Membership in the organization provides identity for all purposes within that organization. Enrollment as member may rely upon another organization that the first one is part of, where the second provides its identity, and so on. The nesting would ordinarily terminate at a self-contained organization (as in Bitcoin) or a real society (as in the banking industry, where regulations require a national identification for each depositor). Certain organizations (for example, social services for drug users) may be self-contained to protect the anonymity of the people they help.

Programming and verifying interactions

Achieving coordination is nontrivial in decentralized applications. Existing approaches hardcode coordination in software implementations. Doing so reduces flexibility in interoperation and hides essential details, thereby preventing the composition of compacts. Blockchains provide coordination abstractions. For example, Hyperledger provides the channel construct: a subnet on which only participants can access information. A channel supports confidentiality and helps decouple participants by hiding irrelevant information. To enable interoperation, we must formalize how an interaction proceeds, not just who participates and what data they exchange. Thus, we face the challenges of how to specify a channel and produce software to interact through a channel.

An effective solution would specify coordination declaratively in conjunction with compacts. Doing

so requires not only formal semantics for data¹¹ but also models of causality and integrity constraints on the interactions underlying the data.¹² In essence, we would formally specify an interaction protocol for each compact that would ensure that the compact could be flexibly enacted, meaning that the protocol would not foreclose any enactment that remained acceptable with respect to the pact. Specifically, the protocol includes a way for each lifecycle state of each norm (including states of satisfaction and violation) in the compact to be realized. To capture the intuition that a decentralized application is specified via a compact, we would need to generate protocols automatically from a compact such that each involved only the relevant principals. The interactions in the protocol would naturally be endowed with a public semantics,² a major benefit of a shared ledger.

The input checker component, realized as a smart contract, in Figure 1, helps ensure the integrity of the information in the blockchain. Thanks to our approach being based on compacts, we can produce the checker's specification from a compact, and the specification can capture, for each principal, the legal actions with respect to the compacts in which that principal participates.

Meaningful trust and reputation

The principals' autonomy in the real world suggests that they would need to trust one another to interoperate. The possibility of a compact's violation creates a vulnerability, a state that constitutes a hallmark of trust.¹³ Blockchain obviates the need for trust only to the extent that the governance structures provide assurance against malfeasance by another and the structures themselves are trusted.

The compensation example illustrates how to achieve coherent interactions without a central authority.^{6,14,15} Governance is a prerequisite for accountability and trust, which are means with which to balance autonomy and correctness.

Blockchain can serve as a platform for promoting meaningful trust. First, quite naturally, the states of relevant compacts provide an opportunity to make evidential trust judgments. The violation and satisfaction of a norm would mean a lowering and raising, respectively, of the trust in the concerned party with respect to similar norms. Second, explicit governance engenders trust by assuring principals that malefactors would be sanctioned. A party may violate a compact by failing to satisfy its conditions; but if it does so, its violation would be determinable from the blockchain. The aggrieved party¹⁹ may file a complaint, which would also be recorded in the blockchain, thereby triggering a governance compact. Third, governance provides a basis for capturing the trust assumptions by formalizing what counts

as evidence for what norm. Consensus on blockchain concerns the events observed. But armed with a governance structure, we can encapsulate norm-relevant evidence within an event to reflect the application meaning. For example, a norm may rely upon a patient having a benign tumor. But in medical practice,²⁰ whether a tumor is benign is a fact that is established by the tumor board of the hospital. That is, the tumor board's assertion counts as the tumor being benign.

PROTOTYPE OVER R3 CORDA

Our compacts-based approach is readily realized on existing blockchains. To illustrate our method, we implemented a proof-of-concept prototype on R3 Corda.²¹ The relevant programming abstractions are as follows: Corda is a network of nodes, each of which hosts a relational database. Each party is a business entity and runs a node. A workflow is a program that composes database transactions and specifies the parties to whom the agreement is visible. By invoking a workflow, a party triggers the transactions in the workflow, which

causes updates to its own database and those of the other parties to whom the matter is visible. An RPCclient is a means by which a party may invoke a workflow programmatically.

Our prototype maps the principal and event constructs to Corda's party and transaction, respectively. It specifies a workflow for each event and encodes agents as RPCclients that invoke the workflows to insert events into the databases of the appropriate parties. The prototype adapts our norms compiler⁷ to generate queries for the lifecycle states of each norm. These queries execute on a node database. Additional details are in the supplementary material.

The emergence of blockchain as a platform for decentralized applications exposes new usage scenarios that bring sociotechnical considerations into computing, specifically, in terms of expectations regarding governance (organizations, norms, and privacy) and trust. Table 1 highlights how our architecture of compacts contrasts with existing approaches. In the compacts approach, the blockchain declaratively represents contractual relationships; maintains relevant events; enables a principal to violate a compact if it so desires; computes whether the compact is satisfied, violated, expired, or pending; and activates the applicable governance compacts, providing a basis for trust.

In this manner, we envision computational representation and reasoning about sociotechnical considerations. Specifically, we advocate developing approaches for programming interactions using blockchain that build on and support effective governance and trust. In this way, compacts differ from the notion of Ricardian contracts,²² which associate

TABLE 1. The differences between compacts and traditional and smart contracts.


	Traditional contracts	Smart contracts	Compacts
Specification	Text	Procedure	Formal, declarative
Automation	None	Full	Compliance checking
Principals' control	Complete	None	Complete
Venue	External	Within blockchain	Recorded on blockchain
Trust model	Hidden	Hardcoded	Explicit
Social meaning	Informal	None	Formal
Standard of correctness	Informal legal	Whatever executes	Formal legal
Scope	Open but ad hoc	Closed	Sociotechnical

ABOUT THE AUTHORS

MUNINDAR P. SINGH is a professor of computer science and codirector of the Science of Security Lablet at North Carolina State University, Raleigh. His research interests include the engineering and governance of sociotechnical systems, including blockchain applications in automating financial transactions. Singh received a Ph.D. in computer sciences from The University of Texas at Austin. He is a former editor-in-chief of *IEEE Internet Computing* and *ACM Transactions on Internet Technology*. He is a Fellow of the IEEE and the Association for the Advancement of Artificial Intelligence. Contact him at singh@ncsu.edu.

AMIT K. CHOPRA is a senior lecturer in the School of Computing and Communications at Lancaster University, United Kingdom. His research interests span sociotechnical systems, multiagent systems, and decentralized systems. Chopra received a Ph.D. in computer science from North Carolina State University, Raleigh. Contact him at amit.chopra@lancaster.ac.uk.

a textual description with a computational one, thereby creating two competing standards of correctness: one that is applied computationally and one that is understood by people. In our approach, there is only one standard. It is high level (understandable by people) and computational (executable by machines).

The compacts-based architecture yields valuable research opportunities concerning how principals 1) preserve autonomy through being able to violate a compact and verify each other's compliance, 2) deal with events in the real business and social worlds that are external to the blockchain, 3) maximize flexibility by having their interactions minimally constrained to interoperate successfully, and most importantly, 4) build and realize governance structures to deal with autonomy and exceptions. 

ACKNOWLEDGMENTS

The work of Munindar P. Singh was funded by an IBM Faculty Award, and the work of Amit K. Chopra was funded by the Engineering and Physical Sciences Research Council [under grant EP/N027965/1 (Turtles)]. We thank Alessandra Scafuro and Samuel Christie for helpful discussions and the anonymous reviewers for their valuable comments.

REFERENCES

1. D. Magazzeni, P. McBurney, and W. Nash, "Validation and verification of smart contracts," *Computer*, vol. 50, no. 9, pp. 50–57, 2017. doi: 10.1109/MC.2017.3571045.
2. M. Singh, "Agent communication languages," *Computer*, vol. 31, no. 12, pp. 40–47, 1998. doi: 10.1109/2.735849.
3. R. Koppel, S. Smith, J. Blythe, and V. Kothari, "Workarounds to computer access in healthcare organizations," in *Driving Quality in Informatics: Fulfilling the Promise*, K. Courtney, A. Kuo, and O. Shabestari, Eds. Amsterdam, The Netherlands: IOS Press, 2015, pp. 215–220.
4. J. dos Santos, J. de Oliveira-Zahn, E. Silvestre, V. da Silva, and W. Vasconcelos, "Detection and resolution of normative conflicts in multi-agent systems," *Autonom. Agents Multi-Agent Syst.*, vol. 31, no. 6, pp. 1236–1282, 2017. doi: 10.1007/s10458-017-9362-z.
5. G. Von Wright, "Deontic logic," *Ratio Juris*, vol. 12, no. 1, pp. 26–38, 1999. doi: 10.1111/1467-9337.00106.
6. M. Singh, "Norms as a basis for governing sociotechnical systems," *ACM Trans. Intelligent Syst. Technol.*, vol. 5, no. 1, pp. 21:1–21:23, 2013. doi: 10.1145/2542182.2542203
7. A. Chopra and M. Singh, "Custard: Computing norm states over information stores," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, 2016, pp. 1096–1105.
8. X. Gao and M. Singh, "Extracting normative relationships from business contracts," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, 2014, pp. 101–108.
9. M. Singh, "An ontology for commitments in multiagent systems," *Artificial Intell. Law*, vol. 7, no. 1, pp. 97–113, 1999. doi: 10.1023/A:1008319631231.
10. M. Singh, A. Chopra, and N. Desai, "Commitment-based service-oriented architecture," *Computer*, vol. 42, no. 11, pp. 72–79, 2009. doi: 10.1109/MC.2009.347.
11. A. Third and J. Domingue, "Linked data indexing of distributed ledgers," in *Proc. World Wide Web Companion*, 2017, pp. 1431–1436.
12. M. Singh, "Semantics and verification of information-based protocols," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, 2012, pp. 1149–1156.
13. C. Castelfranchi and R. Falcone, *Trust Theory*. Hoboken, NJ: Wiley, 2010.
14. J. Pitt and A. Artikis, "The open agent society," *Artificial Intell. Law*, vol. 23, no. 3, pp. 241–270, 2015. doi: 10.1007/s10506-015-9173-y.
15. C. Frantz, M. Purvis, M. Nowostawski, and B. Savarimuthu, "nADICO: A nested grammar of institutions," in *Proc. Int. Conf.*

- Principles and Practice of Multi-agent Systems, LNCS*, vol. 8291. New York: Springer-Verlag, 2013, pp. 429–436.
16. V. Buterin, “Critical update re: DAO vulnerability,” Ethereum, June 17, 2016. [Online]. Available: <https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/>
 17. Hyperledger. Accessed on: Nov. 6, 2019. [Online]. Available: <http://hyperledger.org>
 18. “Compact” in *Longman Dictionary of Contemporary English*. Hoboken, NJ: Pearson English Language Teaching. Accessed on: Nov. 26, 2019. [Online]. Available: <https://www.ldoceonline.com/dictionary/compact>
 19. “Aggrieved party,” *TheLawDictionary.org*. Accessed on: Nov. 26, 2019. [Online]. Available: <http://thelawdictionary.org/aggrieved-party/>
 20. Office of the Assistant Secretary for Planning and Evaluation. (2010). The importance of radiology and pathology communication in the diagnosis and staging of cancer: Mammography as a case study. U.S. Dept. Health and Human Services. Washington, D.C. [Online]. Available: <https://aspe.hhs.gov/system/files/pdf/139361/index.pdf>
 21. R3. Accessed on: Nov. 6, 2019. [Online]. Available: <https://www.r3.com/>
 22. I. Grigg, “The Ricardian contract.” in *Proc. 1st IEEE Int. Workshop Electronic Contracting*, 2004, pp. 25–31. Accessed on: Nov. 6, 2019. [Online]. Available: https://iang.org/papers/ricardian_contract.html

Call for Articles

IEEE Pervasive Computing

seeks accessible, useful papers on the latest peer-reviewed developments in pervasive, mobile, and ubiquitous computing. Topics include hardware technology, software infrastructure, real-world sensing and interaction, human-computer interaction, and systems considerations, including deployment, scalability, security, and privacy.

Author guidelines:
www.computer.org/mc/pervasive/author.htm

Further details:
pervasive@computer.org
www.computer.org/pervasive

IEEE pervasive COMPUTING
 MOBILE AND UBIQUITOUS SYSTEMS

Digital Object Identifier 10.1109/MC.2019.2959871