Digital Social Contracts: A Foundation for an Egalitarian and Just Digital Society*

Luca Cardelli[1], Liav Orgad[2], Gal Shahaf[3], Ehud Shapiro[3], and Nimrod Talmon[4]

[1] University of Oxford luca.a.cardelli@gmail.com
[2] European University Institute liav.orgad@eui.eu
[3] Weizmann Institute {gal.shahaf, ehud.shapiro}@weizmann.ac.il
[4] Ben-Gurion University talmonn@bgu.ac.il

Abstract. Almost two centuries ago Pierre-Joseph Proudhon proposed social contracts – voluntary agreements among free people – as a foundation from which an egalitarian and just society can emerge. A *digital social contract* (DSC) is the novel incarnation of this concept for the digital age: a voluntary agreement between people that is specified, undertaken, and fulfilled in the digital realm. It embodies the notion of "code-is-law" in its purest form, in that a DSC is a program – code in a social contracts programming language, which specifies the digital actions parties to the social contract may take; and the parties to the contract are entrusted, equally, with the task of ensuring that each party abides by the contract. Parties to a social contract are identified via their public keys, and the one and only type of action a party to a DSC may take is a "digital speech act" – signing an utterance with her private key and sending it to the other parties to the contract. We present a formal definition of a DSC as agents that communicate asynchronously via digital speech acts, where the output of each agent is the input of all the other agents. We outline an abstract design for a social contracts programming language and hint on their applicability to social networks, sharing-economy, egalitarian currency networks, and democratic community governance.

Keywords: Democratic Governance · Programming Language Design.

1 Introduction

Over the last three decades, the Internet has mushroomed from 0 to 4.6 billion active "users", 60 per cent of the world population (and more than 90 per cent in the developed world), the fastest diffusion in human history. But what kind of "society" has it created? The digital realm includes a few powerful entities, who

control the entire space, and billions of *data subjects*, as termed by the EU's General Data Protection Regulation (GDPR), who have almost no rights and cannot vote, be consulted, or influence decision-making. They have no due process (e.g., in censorship, account blocking, or privacy violation) and are not the owners of their data. The structure of the digital space remains "feudal" in nature—people are not even perceived as digital "citizens" but as "users"—and it is not based on the ideals on which citizenship in democracies is grounded: liberty, equality, and popular sovereignty. The digital space is further rampant with sybils (fake/duplicate accounts) and bots, making it a hostile environment and untenable for democratic governance. To a large extent, digital technology is the great unequalizer in today's world, making a few digital "barons" rich and powerful by profiting from the multitudes performing digital labour without compensation. Like feudal lords of yore, the digital barons set their platform's rules of conduct (legislative), interpretation (judicial), and implementation (executive), and violate the dictum "no taxation without representation". New technologies do not support the ideals of democracy. Rather, the design of the digital world presents a dystopian future, with the global reach of *surveillance capitalism* [19] robbing citizens of both power and wealth. The structure of the digital world resembles the "state of nature" (i.e., a social disorder).

To establish a *self-governed*, *egalitarian*, and *just society* in the digital world, we advocate the development of DSCs to govern digital human behavior. A DSC is the digital counterpart of social contract theories, as envisioned centuries ago by the founding philosophers of the Enlightenment – Thomas Hobbes, John Locke, Jean-Jacques Rousseau, and Pierre-Joseph Proudhon – and more recent thinkers, such as John Rawls [11, 14, 10, 6, 5]. Social contract theories are the cornerstone of modern democracies (they explain the conditions under which people rationally agree to submit their rights to a legitimate authority in return for collective benefits, such as protection, freedom, or justice) yet, thus far, have remained merely an abstract ideal, based on a hypothetical narrative and the presupposition of tacit, rather than explicit, consent [13]. A DSC offers, for the first time in history, a conceptual model to translate the social contract from a hypothetical theory into a political reality, by utilizing technology in the service of the most important ideas of the Enlightenment. In short, a DSC is a voluntary agreement between individuals, specified, undertaken, and fulfilled in the digital realm. It allows for conditions to reach the social contract to be programmed by generating equal access to information. It further enables the programming of the method of agreement by implementing a "one person, one vote" system, and provides the opportunity to check contractual outcomes in ways that comply with requirements of fairness, e.g., by limiting unequal distributional outcomes. We aim for DSCs that are equal (in sharing power), just (in allocating resources), and self-governed (among the participating members). Hence, the equal sharing of power would require mitigating sybils (fake and duplicate identities) and launching digital currencies where distributed and egalitarian coin minting provides a form of Universal Basic Income [18] to members.

On the technological design, DSCs are programs for a distributed, locally-replicated, asynchronous blockchain architecture. DSCs embody the notion of "code-is-law" in that they are a code in a social-contract programming language, which specifies the digital actions parties may take and the terms. Unlike present digital platforms, this code is not inflicted on the user by a monopolistic digital entity to its financial benefit; rather, people who trust each other enter into a code-regulated relationship voluntarily. DSCs allow for cooperative behavior because they disallow faulty, non-cooperative actions; a party can behave only according to the contract. Agents are expected to employ genuine identifiers and function as both actors and validators. An agent may participate simultaneously in multiple social contracts; similarly, each contract may have a different set of agents as parties; the result is a hypergraph with agents as vertices and contracts as hyperedges. Signed indexed transactions ("digital speech acts"), specified and carried out according to a contract, are replicated only among parties to a contract, who ratify them, and are synchronized independently by each agent. A transaction carried out is finalized when ratified by an appropriate supermajority of the parties to the contract. A party to a contract is typically an agent who embodies a natural person via her genuine identifier, but, if deterministic, could also be a synthetic agent (aka "legal person"), which functions algorithmically, very similar to a "smart contract" of standard blockchains [12].

In 2017, Mark Zuckerberg published a vision on how to turn Facebook from a social network to a "Global Community" governing the planet in most aspects of life, where every user can vote on "global problems that span national boundaries" and "participate in collective decision-making" [20]. According to Zuckerberg, "Facebook can explore examples of how community governance might work at scale" yet, in his global community, people have no rights and bear no responsibilities; they are a means to make profit for Facebook's stakeholders. Zuckerberg's community turns the problem on its head – Facebook is presented as the solution, rather than the problem. Our starting point is radically different than Facebook and similar-minded companies in two aspects: 1) vision: it starts from the premise that there can and should be a digital public life, complementary to the traditional public life in the physical world, by which political communities govern their life; 2) technology: The only technology that provides sovereignty to its community of users is blockchain. Nobody can unplug Bitcoin or Ethereum; both will keep running as long as someone somewhere keeps running their protocols. Alas, current blockchain technologies are environmentally harmful and plutocratic – they are neither egalitarian nor just as sovereignty is shared not equally but according to one's wealth, i.e., one's ability to prove work or stake. We aim to offer an alternative. In summary, DSCs are: **Conceptually**, a voluntary agreement among individuals, specified, undertaken, and fulfilled in the digital realm; **Mathematically**, an abstract model of computation – a transition system specifying concurrent, non-deterministic asynchronous agents engaged in digital speech acts; **Computationally**, a program in a Social-Contracts Programming Language with which social contracts among computational agents are easy to express, comprehend and debug; **Technologically**, a programmable,

distributed, locally-replicated, asynchronous blockchain architecture, operating on mobile phones, tagged by the genuine identifiers of their owners.

Two key concepts are employed in DSCs: (1) **Digital Speech Acts**: Each party to a DSC is equipped with a key pair – a public key and a private key (PKI), with which they can digitally sign text strings and verify each other's signatures. All that a computational agent can do, as a party to a DSC, is perform and perceive digital speech acts, namely sign a digital utterance and send it to the other agents that are parties to the contract, or receive signed digital utterances from these agents. (2) **The Person as an Oracle**: Computational agents in a DSC face nondeterministic choices: Which room to book, when and where? How much to pay and to whom? How to vote on a proposal to accept a new member to the contract? Computational agents have no volition or free will and hence cannot make such choices on their own. Therefore, nondeterministic choices in the social contract code are interpreted as Oracle calls, where the human operating the computational agent serves as the Oracle. In other words, nondeterministic choices in the code of an agent specify the interactions between the "software" and the "user", namely between the computational agent and the person operating it: When faced with a nondeterministic choice, the computational agent consults its human operator as to which choice to make.

Related Work The concepts and design presented here are reminiscent of the notions of blockchains [17], smart contracts [2], and their programming languages [3]. Hand-in-hand with these we are working on egalitarian currency networks [16], an egalitarian and just alternative to existing plutocratic cryptocurrencies such as Bitcoin [7] and Ethereum [3].

A fundamental tenet of our design is that social contracts are made between people who know and trust each other, directly or indirectly via other people that they know and trust. This is in stark contrast to the design of cryptocurrencies and their associated smart contracts, which are made between anonymous and trustless accounts. A challenge cryptocurrencies address is how to achieve consensus in the absence of trust, and their solution is based on proof-of-work [4] or, more recently, proof-of-stake [8] protocols. In contrast, social contracts are between known and trustworthy individuals, each expected to posses a genuine (unique and singular) identifier [15] (see therein discussion on how this can be ensured). Hence, a different approach can be taken. In our approach, the integrity of the ledger of actions taken by the parties to the social contract is preserved internally, among the parties to the agreement, not between external anonymous "miners", as in cryptocurrencies. This gives rise to a much simpler approach to fault tolerance, as discussed in the companion paper [9].

2 Digital Social Contracts

Here we describe a formal model for DSCs – for space constraints, we do not provide full formal definitions for some preliminaries. We assume a given finite set of agents V, each associated with a *genuine* (unique and singular) [15] identifier,

which is also a public key of a key-pair.[5] We expect agents to be realized by computer programs operating on personal devices (e.g. smartphones) of people. Hence, we refer to agents as "it". We identify an agent $v \in V$ with its genuine public identifier, and denote by $v(s)$ the result of agent v signing the string $s \in \mathcal{S}$ with the private key corresponding to v. We assume computational hardness of the public-key system, namely that signatures of an agent with a given identifier cannot be produced without possessing the private key corresponding to this identifier. If t is a tuple indexed by V, then we use $t[v]$ to refer to the v^{th} element of t. We say that $t' = t$, except for $t'[v] := x$, to mean that the tuple t' is obtained from t by replacing its v^{th} element by x. In particular, if $t[v]$ is a sequence, then we say that $t' = t$, except for $t'[v] := t[v] \cdot x$, to mean that t' is obtained from t by appending x to the sequence $t[v]$. Agents are assumed to be connected via a reliable asynchronous communication network (without a message-arrival time limit), and require all messages to be authenticated. Informally, the only things an agent can do as a party to a DSC are (i) perform a *digital speech act* [15], defined next; (ii) observe digital speech acts performed by others; and (iii) change internal state.[6] A key characteristic of a speech act taken in the physical world is that all people present indeed perceive the person taking the act as well as the act itself. We capture this characteristic by requiring that a digital speech act be (i) digitally signed by the person taking it.

Definition 1 (Digital Speech Act, v-act). *Given a set of agents V, a* digital speech act *of agent $v \in V$ consists of (i) signing an utterance (text string) s, resulting in $m = v(s)$; and (ii) broadcasting the message m to V. We refer to a digital speech act by v resulting in the signed action m as the* v-act *m, and let $\mathcal{M}$ be the set of all v-acts for all $v \in V$.*

Definition 2 (Transition System). *A* transition system *$TS = (S, s_0, T)$ consists of a set of states S, an initial state $s_0 \in S$, and a set of transitions T, $T \subseteq S \times S$, with $(s, s') \in T$ written as $s \to s'$. The set $s \to * = \{\hat{s} \mid s \to \hat{s} \in T\}$ is the* outgoing transitions *of s. A* run *of TS is a sequence of transitions $r = s_0 \to s_1 \to \ldots$ from the initial state. A* family *of transition systems $\mathcal{T}(S)$ over S is a set of transition systems (T, S) where T is a set of transitions over S.*

Here we define DSCs in the abstract; in particular, we do not address a distributed realization nor agent faults. These issues are addressed in a companion paper [9]. A DSC consists of a set of agents, identified via public keys, and connected via a reliable, asynchronous, order-preserving communication network. Namely, we assume that between any two agents in the network, the network delivers all messages sent from one agent to another – correctly, eventually, and

[5] We identify the set of agents V with the set of parties to the agreement. Extensions will allow an agent to be a party to multiple agreements, and different agreements to have different sets of agents as parties.

[6] While the formal definition allows a digital speech act to employ an arbitrary string, its intended use is to take meaningful actions. As parties employ strings that are meaningful to the other parties, we believe we do not conjure "speech acts" in vain.

in the order sent.[7] All that agents can do within a DSC is perform digital speech acts (henceforth, *acts*), which are sequentially-indexed utterances, signed by the agent and sent as messages to all other agents that are parties to the contract, as well as receive such messages. Example acts are "I hereby transfer three blue coins to Sue" and "I hereby confirm the room reservation by Joe". A DSC specifies the digital speech acts each party may take at any state. For example, a social contract regarding the blue currency may specify that I can say that I transfer three blue coins to Sue only in a state in which I in fact have at least three blue coins; and I can say that I confirm Joe's room reservation only if I have not already confirmed a conflicting reservation. Our abstract notion of a DSC identifies a DSC with a transition system. The state of a DSC, referred to as a *ledger*, is composed of the states of each of the agents participating in the contract, referred to as their *history*, which is but a sequence of digital speech acts. The history of an agent v consist of digital speech acts it experienced, in particular: acts by v, referred to as *v-acts*, in the order taken, interleaved with acts by the other agents, in the order received from the network. Hence, at any ledger that is a state of the computation, the u-acts in the history of agent v must be a prefix of the u-acts in the history of u, for any u and v. We call such a ledger *sound*. Next we formalizes this informal description. We assume a given set of actions A. Signing an action by an agent v makes the action non-repudiated by v. All that an agent that is party to a DSC does, then, is perform digital speech acts, as well as receive such acts performed by others, resulting in a sequence of non-repudiated acts - a history of crypto speech acts.

Definition 3 (v-act, History). *We refer to $m = v(a)$, the result of signing an action $a \in A$ by agent $v \in V$, as a* v-act*, let $\mathcal{M}$ denote the set of all v-acts by all $v \in V$, and refer to members of $\mathcal{M}$ as* acts*. A* history *is a finite sequence of acts $h = m_0, m_1, \ldots m_n$, $n \in \mathcal{N}$, $m_i = v_i(a_i) \in \mathcal{M}$, $i \in [n]$, $a_i \in A$, $v_i \in V$. The set of all histories is denoted by $\mathcal{H}$.*

Definition 4 (Prefix, Consistent Histories). *Given a sequence $s = x_1, x_2, \ldots x_n$, a sequence s' is a* prefix *of s, $s' \preceq s$, if $s' = x_1, x_2, \ldots x_k$, for some $k \leq n$. $h[u]$ is the restriction of h to the subsequence of u-acts. Two agent histories $h, h' \in \mathcal{H}$ are* consistent *if either $h[v] \preceq h'[v]$ or vice versa for every $v \in V$.*

I.e., histories are consistent if they agree on the subsequences signed by each agent. A ledger of a DSC is an indexed tuple of histories of the parties to the contract. Ledgers are the states of the social contract transition system.

Definition 5 (Ledger). *A* ledger *$l \in \mathcal{L} := \mathcal{H}^V$ is a tuple of histories indexed by the agents V, where l_v is the* history of agent v in l*, or l's* v-history*.*

Following Definition 4, we apply the notation $l_v[u]$ to denote the subsequence of u-acts in the v-history in l. Our key assumption is that in a ledger that is

[7] As agents have public keys with which they sign their sequentially-indexed messages, such a network can be easily realized on an asynchronous network that is not order-preserving and is only intermittently reliable.

a state of a computation, each agent's history is *up-to-date about its own acts*. Therefore, for agent histories to be consistent with each other, the history of each agent can include at most a prefix (empty, partial or complete) of every other agent's acts. In particular , the history of v cannot include u-acts different from the one's taken by u; nor can it run ahead of u and "guess" acts u might take in the future. A ledger is not a linear data structure, like a blockchain, but a tuple of independent agent histories, each of them being a linear sequence of acts. Furthermore, note that l_v, the history of v in ledger l, contains, of course, all v-acts, but it also contains acts by other agents received by v via the communication network. Also, note that the v-history l_v may be behind on the acts of other agents but is up-to-date, or even can be thought of as the authoritative source in l, regarding its own v-acts.

Definition 6 (Ledger Diagonal, Sound Ledger). *Given a ledger $l \in \mathcal{L}$, the* diagonal *of l, l^*, is defined by $l^*_v := l_v[v]$ for all $v \in V$. A ledger is* sound *if $l_u[v] \preceq l^*_v$ for every $u, v \in V$.*

As we assume that each agent is up-to-date about its own acts, the diagonal contains the complete sequence of v-acts for each agent $v \in V$. Thus, each agent history in a ledger reflects the agent's "subjective view" of what actions were taken, where the diagonal of a ledger contains the "objective truth" about all agents. Next we define transitions among these states.

Definition 7 (Transitions). *The set of* social contract transitions $\mathcal{T} \subseteq \mathcal{L} \times \mathcal{L}$ *consists of all pairs $t = l \to l' \in \mathcal{L} \times \mathcal{L}$ where $l' = l$ except for $l'_v := l_v \cdot m$, $m = u(a) \in \mathcal{M}$ for some $u, v \in V$ and $a \in A$. The transition t is referred to as a v-transition, and can also be written as $t = l \xrightarrow{(v,m)} l'$.* □

Definition 8 (Input, Output and Sound Transitions). *A v-transition $t = l \xrightarrow{(v,u(a))} l'$ is* output *if $u = v$ and* input *if $u \neq v$. A transition is sound if it is either input with $l'_v[u] \preceq l_u[u]$ or output.*

Observation 1 *If l is sound and $l \to \hat{l} \in \mathcal{T}$ is sound then $\hat{l}$ is sound.*

The next definition aims to capture the following two requirements: *Output* - an agent v may take a v-act in state l based solely on its own history l_v. In particular, v is free to ignore, or to not know, the other agents' histories in l; *Input* - v must accept messages from the communication network in any proper order the network chooses to deliver them.

Definition 9 (Closed Set of Transitions). *A set of transitions $T \subseteq \mathcal{T}$ is:*

1. Output-closed *if for any v-act m, $v \in V$, $m \in \mathcal{M}$, and any two output transitions $t = l \xrightarrow{(v,m)} \hat{l}$, $t' = l' \xrightarrow{(v,m)} \hat{l}' \in \mathcal{T}$, if $l_v = l'_v$ then $t \in T$ iff $t' \in T$. Namely, if every output v-transition is a function of l_v, independently of l_u for all $u \neq v$.*
2. Input-closed *if T contains every sound input transition in $\mathcal{T}$.*

3. Closed *if it is output-closed and input-closed.*

Definition 10 (Digital Social Contract). *A* DSC *among a set of agents V is a transition system $SC = (S, l_0, T)$ with ledgers as states $S \subseteq \mathcal{L}$, initial state $l_0 := \Lambda^V$, and a closed set of transitions $T \subseteq \mathcal{T}$. The family of all DSCs over S is denoted by $\mathcal{SC}(S)$, and $\mathcal{SC}(\mathcal{L})$ is abbreviated as $\mathcal{SC}$.*

$\mathcal{SC}$ is the family of *abstract social contracts*, in contrast to its more concrete implementations, and view such as specifications for programs in a Social-Contracts Programming Language.

Example 1 (A Currency Community: Example of a DSC). Let $A = \{pay(v)\}_{v \in V}$ be the set of acts. A message $m = u(pay(v))$ corresponds to a payment of 1 coin from u to v. A transition $t = l \xrightarrow{w,u(pay(v))} l'$ records in l_w a payment of 1 coin from u to v. Agent u can record a payment from u to v only if the balance of u, according to l_u, is positive. Formally, let $l_u = m_1, m_2, ..., m_n$, with $m_i = w_i(pay(v_i))$. Set $bal(u) := c + |\{1 \leq i \leq n : \ v_i = u\}| - |\{1 \leq i \leq n : \ w_i = u\}|$, for some $c > 0$, and let $T = \{l \xrightarrow{u,u(pay(v))} l' : \ bal(u) > 0\}$. Let $T \subseteq \tilde{T}$ denote the closure of T in $\mathcal{T}$. The *DSC* $SC = (S, l_0, \tilde{T})$ over the set of agents V specifies a currency community where each agent $u \in V$ is initially granted c coins.

3 A Social Contracts Programming Language

Here we outline the design of a simple social contract programming language. Abstractly, all that an agent can do, as a party to a social contract, is take acts based on its history, and receive acts by others. In a practical social contract, the acts an agent can take typically depend on salient features of its history, rather than on the history in its entirety. So, given that social contracts employ a closed set of transitions, the possible behaviors of an agent can be characterized by a (not necessarily finite and possibly nondeterministic) state transducer. A state transducer reads an input tape (in our case the inputs of an agent v) and writes an output tape (in our case the sequence of v-acts), changing its state in the process. The history of an agent fully specifies, even synchronizes, both tapes.

Hence, our programming language endows each agent with an internal state, which can be viewed as a digest of its history. As in a state transducer, the agent's state may change as a result of an input or an output. Therefore, our proposed *Social-Contracts Programming Language* (SCPL) presents the program of an agent v as a set of rules of the following form:

$$S, m \to m', S', \ \textit{where Conditions.}$$

where S is the internal state (state, for short) of v prior to taking the transition, m is an input u-act by some $u \neq v \in V$, m' is an output v-act, S' is the updated state of v, and *Conditions* are any conditions on S, m, m', and S', that are required for the transition to take place. The intended meaning of such a rule is that an agent in internal state S, upon receiving an act m, may take act m' and

change its internal state to S', if *Conditions* are satisfied. Note that degenerate rules, i.e., rules that do not have input m and/or output m' and/or *Conditions* are allowed. Next we provide an example.

3.1 Egalitarian Governance of Social Contracts

We provide a code example implementing the standard principle of democracy, namely "one person, one vote", in effect showing how a community may form democratically. For simplicity, decisions to add or remove a member are taken by a simple majority, using a secretary that handles ballots, is autonomous, and initiated by the founder.

Program 1: Democratic WhatsApp-like Group

```
founder -->  autonomous#secretary([Self]), member.

secretary(Members), _(propose(X)) -->
    ballot(X,Members,0), secretary(Members).
secretary(Members), ballot(X,[],Result) -->
    secretary_apply(X,Result,Members).

secretary_apply(X,Result,Members) -->
    secretary(Members) where Result =< 0.
secretary_apply(add(Member),Result,Members) ->
    Member#member, secretary([Member|Members])
    where Result > 0.
secretary_apply(remove(Member),Result,Members) -->
    please_leave(Member), secretary(Members')
    where Result > 0,
    Members' is the result of removing Member from Members.

member --> says(X), member.
member, ballot(X,[Self|Members],R) -->
    ballot(X,Members,R'), member
    where R' := R, R+1, or R-1.
member, _(please_leave(Self)) --> says(bye), stop.
member --> says(bye), stop.
```

Note that this is a very simple program, in particular as (1) there is no option to decline an invitation (2) agents must wait for other agents to vote. Indeed, we provide the program mainly as a proof of concept, but to implement a practical democratic decision process, more complex program is needed. We also note that votes are public to the community members and are not anonymous.

Outlook. We have introduced the concept of DSCs, provided a mathematical definition of them, outlined a design for a social contracts programming language, and demonstrated its social utility via program examples. Much remains to be done; some is discussed in companion papers [9, 16].

Acknowledgements

Ehud Shapiro is the Incumbent of The Harry Weinrebe Professorial Chair of Computer Science and Biology. We thank the generous support of the Braginsky Center for the Interface between Science and the Humanities. Nimrod Talmon was supported by the Israel Science Foundation (ISF; Grant No. 630/19).

References

1. Luca Cardelli, Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Digital social contracts: A foundation for an egalitarian and just digital society. *arXiv preprint arXiv:2005.06261*, 2020.
2. Lin William Cong and Zhiguo He. Blockchain disruption and smart contracts. *The Review of Financial Studies*, 32(5):1754–1797, 2019.
3. Chris Dannen. Introducing ethereum and solidity.
4. Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.
5. Thomas Hobbes. *Leviathan*. JM Dent, 1914.
6. John Locke. *Second treatise of government: An essay concerning the true original, extent and end of civil government*. John Wiley & Sons, 2014.
7. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2019.
8. Cong T. Nguyen, Dinh Thai Hoang, Diep N. Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019.
9. Ouri Poupko, Ehud Shapiro, and Nimrod Talmon. Fault-tolerant distributed implementation of digital social contracts. *arXiv preprint arXiv:2006.01029*, 2020.
10. Pierre-Joseph Proudhon and John Beverley Robinson. *General idea of the revolution in the nineteenth century*. Courier Corporation, 2004.
11. John Rawls. *A theory of justice*. Harvard university press, 2009.
12. Wessel Reijers, Fiachra O'Brolcháin, and Paul Haynes. Governance in blockchain technologies & social contract theories. *Ledger*, 1:134–151, 2016.
13. Patrick Riley. *Will and political legitimacy: A critical exposition of social contract theory in Hobbes, Locke, Rousseau, Kant, and Hegel*. Harvard University Press, 2013.
14. Jean-Jacques Rousseau and Gita May. *The social contract: And, the first and second discourses*. Yale University Press, 2002.
15. Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Foundation for genuine global identities. *arXiv preprint arXiv:1904.09630*, 2019.
16. Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Egalitarian and just digital currency networks. *arXiv preprint arXiv:2005.14631*, 2020.
17. Melanie Swan. *Blockchain: Blueprint for a new economy*. O'Reilly Media, 2015.
18. Philippe Van Parijs and Yannick Vanderborght. *Basic income: A radical proposal for a free society and a sane economy*. Harvard University Press, 2017.
19. Shoshana Zuboff. Surveillance capitalism. *Esprit*, 5:63–77, 2019.
20. Mark Zuckerberg. Building global community, 2017.