Volition-Guarded Multiagent Atomic Transactions: Describing People and their Machines

Andy Lewis-Pye
London School of Economics, UK

Ehud Shapiro
London School of Economics, UK, and Weizmann Institute of Science, Israel

Abstract

Formal models for concurrent and distributed systems describe machines; the people who operate them are either ignored or treated as external environment. Yet, key distributed systems—notably grassroots platforms—include people operating their personal machines (smartphones), and their faithful description must include the states of both people and machines and how they jointly effect system behaviour.

Here, we propose volition-guarded multiagent atomic transactions—executed atomically by machines and guarded by their people's volitions—as a mathematical foundation for specifying systems consisting of people operating machines. Each agent's state consists of a volitional state and machine state; a transaction is enabled when the machine precondition holds and the guarding persons are willing. For example, befriending two people is guarded by both; unfriending, by either; a swap of coins is guarded by both parties, while a payment is guarded by the payer.

We develop the mathematical machinery to express safety and liveness of platforms specified in this framework, and provide example specifications of two grassroots platforms: social networks, and currencies. These specifications are then used by AI to derive working implementations. We employ here a simpler definition of 'grassroots' that better captures the informal notion—multiple instances can form and operate independently, yet may coalesce—show that the platforms specified here are grassroots under the new definition, and prove that two instances of a grassroots protocol can always coalesce. We further introduce *volitionally grassroots* protocols, in which two groups can become connected only by mutual consent—the first transaction coupling them must be willed by a member of each—and an agent may leave, never again guarding a transaction, by which a platform records a person's death or incapacity and an AI agent's retirement, two groups each holding an agent that has not left being always able to connect; and show that both platforms are volitionally grassroots.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Software and its engineering → Formal methods

Keywords and phrases Grassroots Protocols, Multiagent Transition Systems, Atomic Transactions, Liveness, Social Networks, Grassroots Currencies, GLP

Best student paper award. This paper is not eligible.

1 Introduction

The gap. Formal models for concurrent and distributed systems describe machines; the people who operate them are either ignored or treated as external environment (see Section 5). Across all surveyed formal traditions—Turing's choice machines [52], CSP [22], CCS [34], I/O automata [32], angelic/demonic nondeterminism [6], ATL [3], game semantics [1], ceremony analysis [18], electronic institutions [19]—the person is modelled as environment, opponent, error source, or unconstrained nondeterministic process, but always as an entity external to the agent, never as a formal component of the agent's state.

Proposal. Here, we propose *volition-guarded multiagent atomic transactions* (henceforth, *volition-guarded transactions*)—executed atomically by machines and guarded by their people's volitions—as a formal foundation for specifying systems consisting of people operating machines. Each agent's state decomposes into a *machine state* and a *volitional state*; a transaction is enabled when the machine precondition holds and the guarding persons are willing. Volitions are thus persistent, inspectable agent state rather than point-of-choice nondeterminism, environment, or opponent moves as in prior formal traditions.

Example: Grassroots social graphs. Each agent p maintains a set of friends $c_p \subseteq P$. The social graph evolves via befriending and unfriending, specified as volition-guarded transactions:

1. **Befriend**: $c'_p := c_p \cup \{q\}$, $c'_q := c_q \cup \{p\}$, provided $q \notin c_p$ and $p \notin c_q$. Guarded by $\{p, q\}$.
2. **Unfriend**: $c'_p := c_p \setminus \{q\}$, $c'_q := c_q \setminus \{p\}$, provided $q \in c_p$ and $p \in c_q$. Guarded by p or q.

Befriending requires both persons to be willing; unfriending can be initiated by either. The guard determines which persons must be willing for a machine transaction to proceed. This distinction is the essence of volition-guarded transactions. A further example, child-safe social networking, appears in Appendix A.

Motivating domain: grassroots platforms. Volition-guarded transactions are a general framework for systems of people operating machines. An example of a class of such systems, which motivated this work, is grassroots platforms [41, 42, 44, 49], which aim to offer an egalitarian alternative to global platforms, centralized and decentralized alike. Grassroots platforms consist of people operating their personal machines (smartphones), and can have multiple instances that emerge and operate independently of each other and of any global resource except the network, yet can interoperate and coalesce once interconnected. Key grassroots platforms include grassroots social networks [42, 48], grassroots coins and bonds [28, 44], and grassroots democratic communities [20, 49, 23]. No platform that operates on a shared global resource—a replicated ledger (Blockchain [36]), a distributed data structure (IPFS [7], DHT [33]), or a distributed pub/sub system with a global directory [13, 14, 11])—is grassroots. Mastodon [37] is peer-to-peer among servers, not people; BitTorrent [51] requires reaching a tracker or DHT boot nodes to join a swarm; Scuttlebutt [50, 24] is grassroots in design even if not formally proven as such.

An earlier paper [45] introduced multiagent atomic transactions for grassroots platforms and presented transactions-based specifications of social graphs, cryptocurrencies, and democratic federations; that work did not address the role of people, transaction equivalence, or liveness, and employed the original definition of grassroots protocols.

Framework overview. We develop the mathematical machinery needed to express volition-guarded transactions and the safety and liveness of grassroots platforms they specify. We introduce *transaction equivalence classes*, which provide a natural notion of "same action" across configurations, and define liveness in terms of such classes: a run is live iff every persistently enabled class is eventually taken. We demonstrate the framework with two grassroots platforms:

1. **Grassroots social networks** [42, 48] let people maintain their friendship connections through local storage and peer-to-peer relationships without central control, with the social graph evolving through befriending and unfriending.
2. **Grassroots currencies** [28, 44] let each person mint their own coins, backed by the goods and services they offer, and exchange them with others via atomic swaps; bonds extend coins with a maturity date, enabling interest-bearing credit, loans, and the full gamut of financial instruments, and we specify the coins.

The specifications presented here have been used by AI to derive working implementations in GLP, a grassroots multiagent concurrent logic programming language [43], as reported in [46, 48, 44, 47]. We employ here a simpler definition of grassroots protocols than the original [41, 45], based on interleavings of correct runs, which better captures the informal notion of grassroots, show that the platforms specified here are grassroots under the new definition, and prove that two instances of a grassroots protocol can always coalesce. We define *volitionally grassroots* protocols, in which two groups can become connected only by the mutual consent of a member of each—the first transaction coupling them is guarded on both sides—and in which an agent may leave, never again guarding a transaction, by which a platform records a person's death or incapacity and an AI agent's retirement, two groups each holding an agent that has not left being always able to connect; and prove both platforms volitionally grassroots. We also prove that Bitcoin is not grassroots and show that the argument applies to distributed hash tables and IPFS.

Use of AI tools. AI tools were used in drafting and revising the text of this paper. The working implementations derived by AI from the specifications presented here are reported in Appendix E.

Paper outline. Section 2 introduces the formal framework: volition-guarded transactions and the volitional transactions they induce, transaction equivalence, and liveness. Section 3 defines protocols and grassroots protocols via interleavings of correct runs, gives a containment condition and a syntactic guard condition under which a transactions-based protocol is grassroots, proves that two instances of a grassroots protocol can always coalesce, and defines volitionally grassroots protocols, in which every first interaction between two groups is willed by a member of each and an agent may leave. Section 4 presents two grassroots platforms (social networks, currencies), proves their safety properties, and proves that they are volitionally grassroots. Section 5 discusses related work. Section 6 concludes and discusses future work. The appendix contains further examples of grassroots platforms (Appendix A), proofs deferred from the body (Appendix B), the proof that Bitcoin is not grassroots and its application to distributed hash tables and IPFS (Appendix C), the machinery of implementations, faults, and resilience (Appendix D), notes on implementation (Appendix E), and a detailed survey of formal models of persons in concurrent systems (Appendix F).

2 Volitional Multiagent Transition Systems

Earlier work introduced multiagent transition systems [39], grassroots protocols and platforms [41], and their definition via multiagent atomic transactions [45]—capturing the

behaviour of machines but not of the people operating them. Here we develop the mathematical machinery to describe agents consisting of a person and a machine, and the volitional transactions they execute.

A volitional transaction is a "regular" multiagent atomic transaction—henceforth, a *machine transaction*—guarded by the volitions of some, all, or none of the people whose machines participate; a person's volitional state is a set of equivalence classes of machine transactions they are willing their machine to participate in. The social graph illustrates the two extremes: befriending is guarded by both p and q (the class of the 'befriend p and q' machine transaction must be in both volitions), while unfriending is guarded by either. A person may freely change their volitional state via volition changes, and may act at their own machine by inputs; additionally, an equivalence class is removed from every agent's volitional state when a machine transaction in this class is taken, fulfilling the will.

The definitions of volitional transactions and agent states are mutually recursive: a volitional transaction is a machine transaction guarded by volitions, which are themselves sets of equivalence classes of machine transactions. We resolve this circularity bottom-up: first machine transactions, then their equivalence, then agent states (including volitional states), and finally volitional transactions.

2.1 Agents, People, Machines, Volition-Guarded Transactions and Volitional Transactions

We assume a countable, potentially infinite set of *agents* Π, but consider only finite subsets of it, so when referring to a particular set of agents $P \subset \Pi$ we assume P to be nonempty and finite. We use $\subset$ to denote the strict subset relation and $\subseteq$ when equality is also possible, and use $p \neq q \in P$ as a shorthand for $p \in P \wedge q \in P \wedge p \neq q$. As standard, we use S^P to denote the set of all total functions from P to S, and if $c \in S^P$ we use c_p (instead of $c(p)$) to denote the value of c at $p \in P$.

▶ **Definition 1** (Machine State, Configuration, Transaction, Volition-Guarded Multiagent Atomic Transaction). *Given an arbitrary set S of **machine states**, with a designated **initial state** $s_0 \in S$, and agents $Q \subset \Pi$, a **machine configuration** over Q is a member of S^Q, and a **machine transaction** over **participants** Q is a pair $c \to c' \in (S^Q)^2$ such that $c \neq c'$; it is **unary** if $|Q| = 1$. Given such a machine transaction t, a **volition-guarded multiagent atomic transaction** over t—henceforth, **volition-guarded transaction**—is a pair (t, Q') where $Q' \subseteq Q$ are its **guards**.*

Machine transactions are atomic and asynchronous [39]—they can be carried out by their participants at any time, regardless of the states of non-participants. Participants include both active agents (whose state changes) and stationary agents (whose state is a precondition but does not change). Volition-guarded transactions are machine transactions that can be carried out only if their guards $Q' \subseteq Q$ are willing. Volition-guarded transactions do not distinguish between agents that initiate a transaction and those willing to participate in it. When we say a transaction is "guarded by $\{p, q\}$," both must be willing; when we say it is "guarded by either p or q," we mean there are two volition-guarded transactions over the same machine transaction, $(t, \{p\})$ and $(t, \{q\})$, so that either person's volition suffices.

Distinct machine transactions can represent "the same action" in different configurations; we capture this with an equivalence relation on machine transactions.

▶ **Definition 2** (Transaction Equivalence). *Given a set of machine transactions R, a **transaction equivalence** is an equivalence relation $\sim$ on R such that $t \sim t'$ implies t and t' have the same participants, and $R/\sim$ is countable. We write $[t]$ for the equivalence class of t under $\sim$.*

For example, all befriend(p, q) transactions—differing only in the configurations in which they occur—form an equivalence class. Further examples for each platform appear in Section 4.

Next we provide the mathematical machinery allowing people to express their volitions regarding the classes of machine transactions their machines may participate in.

▶ **Definition 3** (Agent State and Configuration). *Given agents P, states S with initial state s_0, a set of machine transactions T each over its own participants $Q \subseteq P$ and S, and equivalence $\sim$ on T, an **agent state** is a pair $(V, m) \in \mathcal{A} = (2^{T/\sim} \times S)$ where V is its **volitional state** and $m \in S$ its **machine state**. The **initial agent state** is $(\emptyset, s_0)$. An **agent configuration** c over P, S, T, and $\sim$ is a member $c \in \mathcal{A}^P$ in which $c_p^v \subseteq (T/\sim)_p$ for every $p \in P$, where $(T/\sim)_p$ denotes the classes in $T/\sim$ in which p is a participant; we write c_p^v for the volitional state and c_p^m for the machine state of agent p in c.*

▶ **Definition 4** (Volitional Transaction). *Given agents P, states S, machine transactions T over P and S, and equivalence $\sim$ on T:*

1. *A **volition change of agent** $p \in P$ is a pair $c \to c'$ of agent configurations over $\{p\}$, S, T, and $\sim$ such that $c_p^v, {c'}_p^v \subseteq (T/\sim)_p$ and $c_p^v \neq {c'}_p^v$, and $c_p^m = {c'}_p^m$.*
2. *A **volitional machine transaction** induced by a volition-guarded transaction (t, Q'), for some $t = (d \to d') \in T$ over $Q' \subseteq Q \subseteq P$, is a pair $c \to c'$ where $c \neq c'$ are agent configurations over P, S, T, and $\sim$ such that $[t] \in c_q^v$ for every $q \in Q'$; $c_p^m = d_p$ and ${c'}_p^m = d'_p$ for every $p \in Q$; $c_p^m = {c'}_p^m$ for every $p \in P \setminus Q$; and ${c'}_p^v = c_p^v \setminus \{[t]\}$ for every $p \in P$.*
3. *A **volitional transaction** is a volition change or a volitional machine transaction.*

When a volitional machine transaction induced by (t, Q') is taken, the class $[t]$ is removed from every agent's volitional state: the will is fulfilled by any equivalent transaction. A person may independently change their volitional state via volition changes, which may add or remove classes; beyond these, the framework removes a class from c_p^v only upon fulfilment.

A person's not willing a transaction has, so far, no consequence: no volitional transaction is enabled by the absence of a will. An implementation of volition-guarded transactions by communicating agents [48] needs one, so that a machine asked for its person's will answers either way; it is the following, by which the machine answers for a person who does not will an act: in machine state s, if p wills it s' is taken, else s''.

▶ **Definition 5** (Volition-Guarded If-Then-Else Transaction, Then-Branch, Else-Branch). *Given machine states S, a **volition-guarded if-then-else transaction** over a single participant $p \in \Pi$ is a triple $s \to s'\,;\, s''$ **guarded by** p, with $s, s', s'' \in S$, $s \neq s'$ and $s \neq s''$; the unary machine transaction $s \to s'$ is its **then-branch** and $s \to s''$ its **else-branch**.*

▶ **Definition 6** (Else Transaction). *Given agents P, states S, machine transactions T over P and S, and equivalence $\sim$ on T, an **else transaction** induced by a volition-guarded if-then-else transaction $s \to s'\,;\, s''$ guarded by $p \in P$, with $s \to s' \in T$, is a pair $c \to c'$ of agent configurations over P, S, T, and $\sim$ such that $[s \to s'] \notin c_p^v$; $c_p^m = s$ and ${c'}_p^m = s''$; $c_q^m = {c'}_q^m$ for every $q \in P \setminus \{p\}$; and ${c'}_q^v = c_q^v$ for every $q \in P$.*

The then-branch, taken when p wills it, is the unary volition-guarded transaction $(s \to s', \{p\})$ and induces volitional machine transactions per Definition 4; a unary volition-guarded transaction is thus the special case with no else-branch. The else transaction is taken when p does not will the then-branch; it fulfils no will and removes no class. The two are placed in one class below, so that liveness obliges the machine to answer either way; the obligation is the machine's, and it has no time bound. Only a unary transaction has an else-branch: a binary one would let an agent that never wills it trigger it (Section 3.3).

A person also acts at their own machine by means other than willing: naming an agent, entering a text or a number. Such an act is an input, a unary machine transaction of the agent that is taken when the person makes it and is in no class, so that no run is obliged to take it.

▶ **Definition 7** (Input, Input Transition). *Given machine states S, an **input** of agent $p \in \Pi$ is a unary machine transaction $s \to s'$ over $\{p\}$. Given agents P, states S, machine transactions T over P and S, and equivalence $\sim$ on T, an **input transition** of $p \in P$ induced by an input $s \to s'$ of p is a pair $c \to c'$ of agent configurations over P, S, T, and $\sim$ such that $c_p^m = s$ and $c'^m_p = s'$; $c_q^m = c'^m_q$ for every $q \in P \setminus \{p\}$; and $c'^v_q = c_q^v$ for every $q \in P$.*

2.2 Induced Transition Systems

The following is a simplified variation, sufficient for the purpose of this work, on the foundations introduced in [39].

▶ **Definition 8** (Transition System, Computation, Run, Safe, Live, Correct). *A **transition system** is a tuple $TS = (S, s_0, T, \sim)$, where:*

1. *S is an arbitrary non-empty set, referred to as the set of **states**.*
2. *Some $s_0 \in S$ is the designated **initial state**.*
3. *$T \subseteq S^2$ is a set of **correct transitions over** S, where each transition $t \in T$ is a pair (s, s') of non-identical states $s \neq s' \in S$, also written as $t = s \to s'$.*
4. *$\sim$ is a **partial equivalence relation** on T: a symmetric and transitive relation on T, not necessarily reflexive. Its domain $\{t \in T : t \sim t\}$ is partitioned into countably many **liveness classes** $T/\sim$; a transition outside the domain belongs to no class.*

*A **computation** of TS is a (nonempty, potentially infinite) sequence of states $r = s_1, s_2, \cdots$; it is a **run** of TS if $s_1 = s_0$. A **prefix** of a computation is a finite initial segment of it. A computation $r = s_1, s_2, \ldots$ is **safe**, also written $r \subseteq T$, if $s_i \to s_{i+1} \in T$ for every two consecutive states; and $s \xrightarrow{*} s' \subseteq T$ denotes the existence of a safe computation from s to s' (empty if $s = s'$). A class $[t] \in T/\sim$ is **enabled** in a state s if $s \to s' \in [t]$ for some $s' \in S$. A run r is **live** if no class $[t] \in T/\sim$ is enabled in every state of some suffix of r with no member of $[t]$ occurring in the suffix. A run is **correct** if it is safe and live.*

A partial equivalence, rather than a total one, is used so that some transitions may carry no liveness requirement at all: only transitions in the domain of $\sim$ form classes and thereby incur a liveness obligation, while transitions outside the domain—belonging to no class—may occur in a correct run but are never required to. A run with uncountably many classes enabled in every state of a suffix is not live.

▶ **Lemma 9** (Extension). *Every finite safe run of a transition system is a prefix of a correct run of it.*

▶ **Definition 10** (Multiagent Transition System). *Given agents $P \subset \Pi$ and an arbitrary set S of **states** with a designated **initial state** $s_0 \in S$, a **multiagent transition system** over P and S is a transition system $TS = (C, c_0, T, \sim)$ with **configurations** $C := S^P$, **initial configuration** $c_0 := \{s_0\}^P$, **transitions** $T \subseteq C^2$ a set of transactions over P and S, and $\sim$ a partial equivalence on T.*

Unary multiagent transition systems were introduced in [39] and were employed to define the notion of grassroots protocols [41] and to provide unary specifications for various grassroots platforms [42, 40, 28]. Here, we employ k-ary transition systems, for any $k \leq |P|$, in which several agents can change their state simultaneously.

Rather than specifying a multiagent transition system over a set of agents P directly, we specify it via machine transactions (Definition 1).

A machine transaction over $Q \subseteq P$ defines a set of multiagent transitions over P in which all members of $P \setminus Q$ are stationary:

▶ **Definition 11** (Transaction Closure). *Let $P \subset \Pi$, S a set of machine states, and $C := S^P$. For any transition or transaction $t = c \to c'$, we write $t_q := c_q \to c'_q$ and say p is **stationary** in t if $c_p = c'_p$. For a machine transaction $t = (c \to c')$ over S with participants Q, the **P-closure of** t, $t{\uparrow}P$, is the set of transitions over P and S defined by:*

$$t{\uparrow}P := \begin{cases} \{t' \in C^2 : \forall q \in Q.(t_q = t'_q) \wedge \forall p \in P \setminus Q.(p \text{ is stationary in } t')\} & \text{if } Q \subseteq P \\ \emptyset & \text{otherwise} \end{cases}$$

*If R is a set of machine transactions, each $t \in R$ over some Q and S, then the **P-closure of** R, $R{\uparrow}P$, is the set of transitions over P and S defined by:*

$$R{\uparrow}P := \bigcup_{t \in R} t{\uparrow}P$$

*Given a relation $\sim$ on R, its **P-closure** $\sim{\uparrow}P$ is the relation on $R{\uparrow}P$ with $\hat{t}$ $(\sim{\uparrow}P)$ $\hat{t}'$ iff $\hat{t} \in t{\uparrow}P$ and $\hat{t}' \in t'{\uparrow}P$ for some $t \sim t'$. A transition over P is **unary** if it lies in the P-closure of a unary transaction.*

Namely, the closure over $P \supseteq Q$ of a machine transaction t over Q includes all transitions t' over P in which members of Q do the same in t and in t', and the rest remain in their current (arbitrary) state. The closure likewise carries any relation on transactions to one on the induced transitions. The closure thus turns a transaction over its participants into transitions over all of P, and the equivalence groups transitions that are one act. If distinct transactions in R have disjoint P-closures—as when every participant of every transaction in R changes state—then each transition in $R{\uparrow}P$ has a unique inducing transaction, $\sim{\uparrow}P$ relates two transitions exactly when their inducing transactions are $\sim$-related, and $\sim{\uparrow}P$ is a partial equivalence whenever $\sim$ is. Disjointness is a condition on R: a participant that does not change state, such as an agent whose state a transaction reads, is allowed provided no other transaction in R moves the same agents in the same way. For volition-guarded transactions it is required only of unguarded ones (Definition 12).

A set R of volition-guarded transactions with an equivalence $\sim$ on their underlying machine transactions, and a set I of inputs, induce a volitional multiagent transition system:

▶ **Definition 12** (Volitional Multiagent Transition System). *Let $P \subset \Pi$ be agents and S machine states with initial state s_0. Let R be a set of volition-guarded transactions and volition-guarded if-then-else transactions, a volition-guarded if-then-else transaction $s \to s'\,;\, s''$ guarded by p standing in R for its then-branch $(s \to s', \{p\})$ together with its else-branch, and I a set of inputs of agents of P. Let $T_R := \{t : (t, Q') \in R \text{ for some } Q'\}$ be the underlying machine transactions, and $\sim$ an equivalence on T_R, hence a partial equivalence in the sense of Definition 8. Assume that every participant of every transaction in R is in P; that two machine transactions of different classes, each the machine transaction of an unguarded transaction in R or an else-branch, have disjoint P-closures; that no else-branch and no input is in T_R; and that volition-guarded if-then-else transactions with the same else-branch have $\sim$-equivalent then-branches. The **volitional multiagent transition system induced by** $(S, R, I, \sim)$ **over** P is the multiagent transition system $(\mathcal{A}^P, c_0, T_V, \sim_V)$ where:*

1. *$\mathcal{A} := 2^{T_R/\sim} \times S$ is the **agent state space**;*

2. $c_0 \in \mathcal{A}^P$ *is the* ***initial agent configuration****, with* $c_{0p}^v = \emptyset$ *and* $c_{0p}^m = s_0$ *for every* $p \in P$*;*
3. T_V *consists of all transitions* $e \to e' \in (\mathcal{A}^P)^2$ *of one of four forms: (i) a* ***volition change*** *of some* $p \in P$*—*$e_p^v, e'^v_p \subseteq (T_R/\!\sim)_p$ *and* $e_p^v \neq e'^v_p$*,* $e_p^m = e'^m_p$*, and* $e_r = e'_r$ *for every* $r \in P \setminus \{p\}$*; (ii) a* ***volitional machine transaction*** *induced by some volition-guarded transaction* $(t, Q') \in R$ *per Definition 4(2); (iii) an* ***else transaction*** *induced by some volition-guarded if-then-else transaction in* R *per Definition 6; or (iv) an* ***input transition*** *of some* $p \in P$ *induced by some input in* I *per Definition 7;*
4. $\sim_V$ *relates two volitional machine transactions or else transactions whenever their inducing machine transactions are* $\sim$*-equivalent, where the inducing machine transaction of a volitional machine transaction induced by* (t, Q') *is* t *and that of an else transaction is the then-branch* $s \to s'$ *of its volition-guarded if-then-else transaction, and leaves every volition change and input transition outside its domain—so volition changes and input transitions belong to no class and carry no liveness obligation.*

Every transition of T_V other than a volition change or an input transition has an inducing machine transaction, and $\sim$ is an equivalence on T_R, so $\sim_V$ is a partial equivalence with countably many classes, and two $\sim_V$-related transitions have the same participants. Every configuration reachable from c_0 is an agent configuration (Definition 3), each agent holding only classes it participates in; a member of $\mathcal{A}^P$ that is not one is reached by no run. A step induced by a guarded transaction (t, Q'), $Q' \neq \emptyset$, removes $[t]$ from the volitional state of each guard, which held it; a step of another class removes another class, and an else transaction removes none, so neither is that step, and two guarded transactions of different classes may share their P-closures. Two unguarded transactions of different classes are kept apart by the condition on P-closures. The two conditions on else-branches give each else transaction one class: an else-branch that is not a transaction of R is induced only by if-then-else transactions, and those sharing it have $\sim$-equivalent then-branches. The set R of volition-guarded transactions that specifies a platform thus determines the volitional machine transitions of the induced volitional multiagent transition system, while volition-change transitions are freely available to every agent independent of R, and input transitions are available to each agent as I says.

We define when a volition-guarded transaction is enabled, combining the machine precondition with the volitions of its guards.

▶ **Definition 13** (Enabled)**.** *Given a set of volition-guarded transactions, each* (t, Q') *with* $t = d \to d'$ *a machine transaction over some* $Q' \subseteq Q \subseteq P$ *and* S*, and an equivalence* $\sim$ *on machine transactions: the volition-guarded transaction* (t, Q') *is* ***machine-enabled*** *in agent configuration* c *over* P *if* $c_p^m = d_p$ *for every* $p \in Q$*, and* ***enabled*** *if in addition* $[t] \in c_q^v$ *for every* $q \in Q'$*. The else-branch of a volition-guarded if-then-else transaction* $s \to s'\,;\, s''$ *guarded by* p *is* ***enabled*** *in* c *if* $c_p^m = s$ *and* $[s \to s'] \notin c_p^v$*. An equivalence class* $[t]$ *is* ***enabled*** *in* c *if some volition-guarded transaction* (t', Q') *with* $t' \in [t]$ *is enabled in* c*, or the else-branch of some volition-guarded if-then-else transaction with then-branch* $t' \in [t]$ *is enabled in* c*.*

A volition-guarded transaction with an empty guard ($Q' = \emptyset$) requires no volitions and is enabled whenever its machine precondition is met. The class of the then-branch of a volition-guarded if-then-else transaction guarded by p is thus enabled whenever p's machine is in its source state s: by its then-branch if p wills it, and by its else-branch if not. For the volitional multiagent transition system above, a class of volitional machine transactions and else transactions is enabled at a configuration in the sense of Definition 8 exactly when some volition-guarded transaction or else-branch inducing it is enabled in the above sense; this determines which runs are live and correct per Definition 8. Volition-change transitions and

input transitions belong to no class and so impose no liveness obligation; personal choices remain free.

3 Grassroots Protocols

Here we define what a protocol is; define when a protocol is grassroots, using the notion of interleaving of correct runs; show how to define a protocol via a set of transactions; prove that any transactions-based protocol is oblivious under a natural condition; conclude that if it is also interactive, it is grassroots; and show that two instances of a grassroots protocol can always coalesce. For volitional transactions-based protocols, guarding every transaction of two or more participants guarantees this condition; and the volitional notion, volitionally grassroots, requires the first interaction between any two groups to be willed by a member of each, and interactivity only of groups each holding an agent that has not left. Proofs omitted from the body are in Appendix B.

3.1 Protocols and Grassroots Protocols

A protocol is a family of multiagent transition systems, one for each set of agents $P \subset \Pi$, which share an underlying set of local machine states $\mathcal{S}$ with a designated initial state s_0. A *local-states function* maps every set of agents $P \subset \Pi$ to an arbitrary set of local machine states $S(P) \subset \mathcal{S}$ that includes s_0 and satisfies $P \subset P' \subset \Pi \implies S(P) \subset S(P')$.

▶ **Definition 14** (Protocol). *A **protocol** $\mathcal{F}$ over a local-states function S is a family of multiagent transition systems that has exactly one transition system $\mathcal{F}(P) = (C(P), c_0(P), T(P), \sim(P))$ for every $P \subset \Pi$, with **agent states** $\mathcal{A}(P)$, configurations $C(P) := \mathcal{A}(P)^P$, initial configuration $c_0(P) \in C(P)$, and partial equivalence $\sim(P)$ on $T(P)$ determined by the protocol, such that $P \subseteq P' \subset \Pi$ implies $\mathcal{A}(P) \subseteq \mathcal{A}(P')$ and $c_0(P)_p = c_0(P')_p$ for every $p \in P$.*

Informally, in a grassroots protocol two disjoint groups of agents can each operate independently—their interleaved correct runs are correct runs of the combined system—yet the combined system offers genuinely new behaviours that neither group could produce on its own. To capture this notion formally, we first define the interleaving of runs of two disjoint groups.

▶ **Definition 15** (Interleaving). *Let $P, P' \subset \Pi$ be disjoint nonempty sets of agents, $r = c_0, c_1, \ldots$ a run of $\mathcal{F}(P)$, and $r' = d_0, d_1, \ldots$ a run of $\mathcal{F}(P')$. An **interleaving** of r and r' is a sequence $e_0, e_1, \ldots$ of configurations in $C(P \cup P')$ for which there exist non-decreasing sequences of indices $(i_k)_{k\geq 0}$ and $(j_k)_{k\geq 0}$ with $i_0 = j_0 = 0$ such that for every $k \geq 0$:*

1. *$(e_k)_p = (c_{i_k})_p$ for every $p \in P$,*
2. *$(e_k)_q = (d_{j_k})_q$ for every $q \in P'$,*
3. *if e_{k+1} exists, then exactly one of: (i) $i_{k+1} = i_k + 1$ and $j_{k+1} = j_k$ (a P-step), or (ii) $i_{k+1} = i_k$ and $j_{k+1} = j_k + 1$ (a P'-step).*

Moreover, if r is finite of length n then $i_k = n$ for some k, and if r is infinite then for every $m \geq 0$ there is a k with $i_k = m$; likewise for r' and (j_k).

Note that an interleaving is well-defined: by Definition 14, $\mathcal{A}(P) \subseteq \mathcal{A}(P \cup P')$ and $\mathcal{A}(P') \subseteq \mathcal{A}(P \cup P')$, so each e_k, with p-components in $\mathcal{A}(P)$ and q-components in $\mathcal{A}(P')$, is a valid configuration in $C(P \cup P') = \mathcal{A}(P \cup P')^{P \cup P'}$. Also, $e_0 = c_0(P \cup P')$, since $(e_0)_p = (c_0)_p = c_0(P \cup P')_p$ for $p \in P$ and $(e_0)_q = (d_0)_q = c_0(P \cup P')_q$ for $q \in P'$, by the agreement of initial configurations across $\mathcal{F}(P)$, $\mathcal{F}(P')$, and $\mathcal{F}(P \cup P')$.

▶ **Definition 16** (Interaction, Interactive Run, First Interaction). *Let $Q \subset \Pi$. For a configuration c over Q and $R \subseteq Q$, write $c|_R$ for the restriction of c to R. A transition $e \to e'$ of $\mathcal{F}(Q)$ is an **interaction between** $x \neq y \in Q$ if $e|_{\{x\}} \neq e'|_{\{x\}}$ and $e|_{\{y\}} \neq e'|_{\{y\}}$, and $e|_{\{x\}} \to e'|_{\{x\}}$ is not a transition of $\mathcal{F}(\{x\})$ or $e|_{\{y\}} \to e'|_{\{y\}}$ is not a transition of $\mathcal{F}(\{y\})$. It is an **interaction between** disjoint nonempty $P, P' \subseteq Q$ if it is an interaction between some $x \in P$ and some $y \in P'$. A run $\hat{r}$ of $\mathcal{F}(Q)$ is **interactive between** P **and** P' if some transition of $\hat{r}$ is an interaction between them; the **first interaction** of P and P' in $\hat{r}$ is the first such transition.*

We can now define the key notion of this paper, a grassroots protocol. The following definition improves upon the original definition [41], which was formulated in terms of a subset relation ($P \subset P'$) and conditions on the availability of transitions, and upon the definition of [45], which did not incorporate liveness. The new definition captures the informal notion of grassroots directly, using disjoint groups and the interleaving of their correct runs; the differences are discussed below.

▶ **Definition 17** (Oblivious, Interactive, Grassroots). *A protocol $\mathcal{F}$ is:*

1. ***oblivious** if for every disjoint nonempty $P, P' \subset \Pi$, every interleaving of a correct run of $\mathcal{F}(P)$ and a correct run of $\mathcal{F}(P')$ is a correct run of $\mathcal{F}(P \cup P')$.*
2. ***interactive** if for every $Q \subset \Pi$ and disjoint nonempty $P, P' \subseteq Q$, every prefix of a correct run of $\mathcal{F}(Q)$ containing no interaction between P and P' is a prefix of a correct run of $\mathcal{F}(Q)$ that contains one.*
3. ***grassroots** if it is oblivious and interactive.*

Oblivious. Being oblivious means two disjoint groups coexist without interference: any interleaving of a correct run of one with a correct run of the other is a correct run of the combined system, so each group's correct operation is undisturbed by the other's presence.

Interactive. Being interactive means two groups that have not yet interacted can still do so: the combined system has a correct continuation in which a single transaction engages agents of both groups—a coupling that no interleaving of the groups' separate runs can produce. The condition is required at every such point and not at the initial configuration alone, so a protocol whose groups can reach a state from which they can never couple is not interactive. The first such transaction in a run is its first interaction (Definition 16). The substance of interactivity is the content of these transactions, which the platforms in Section 4 illustrate.

Coalescence. Agents that have taken part in a common interaction are joined; an instance is a maximal set of agents joined directly or indirectly, and two instances coalesce when they come to be one.

▶ **Definition 18** (Interaction Graph, Instance, Coalescence). *Let $\mathcal{F}$ be a protocol, $Q \subset \Pi$, and r a prefix of a run of $\mathcal{F}(Q)$. The **interaction graph** of r has vertices Q and an edge between $a \neq b \in Q$ whenever some transition of r is an interaction between a and b. A nonempty $P \subseteq Q$ is an **instance** in r if it is a connected component of the interaction graph of r. Disjoint nonempty $P, P' \subseteq Q$ **coalesce** in a run $\hat{r}$ of $\mathcal{F}(Q)$ if they are instances in some prefix of $\hat{r}$ and lie in one component of the interaction graph of a longer prefix of $\hat{r}$.*

▶ **Proposition 19** (Coalescence). *Let $\mathcal{F}$ be a grassroots protocol, $Q \subset \Pi$, and $P, P' \subseteq Q$ disjoint instances in a prefix r of a correct run of $\mathcal{F}(Q)$. Then r is a prefix of a correct run of $\mathcal{F}(Q)$ in which P and P' coalesce.*

Proof. Take $p \in P$ and $q \in P'$. As P and P' are distinct components of the interaction graph of r, no transition of r is an interaction between p and q. By interactivity, r is a prefix

of a correct run of $\mathcal{F}(Q)$ some transition of which is such an interaction; the interaction graph of that run has the edge between p and q, so P and P' lie in one of its components. ◀

In Appendix C we show that federated systems (Mastodon) are not interactive, and that systems employing a shared global data structure—Bitcoin, distributed hash tables, IPFS, trackerless BitTorrent—are not oblivious, hence none of these is grassroots.

3.2 Transactions-Based Grassroots Protocols

A protocol can be specified by a set of machine transactions over a local-states function, lifting the closure construction of Section 2 from a fixed set of agents to the whole family. We show that such a protocol is oblivious under a natural condition, and hence grassroots when it is also interactive.

▶ **Definition 20** (Transactions Over a Local-State Function). *Let S be a local-states function. A set of transactions R is **over** S if every transaction $t \in R$ is a multiagent transition over Q and $S(P')$ for some $Q \subseteq P' \subset \Pi$. Given such a set R and $P \subset \Pi$, $R(P) := \{t \in R : t$ is over Q and $S(P'), Q \subseteq P' \subseteq P\}$.*

A transaction of agents of P from local states of P may reach a local state of a larger set of agents, as when p adds to its state an agent outside P. Such a transaction is in $R(P \cup P')$ and not in $R(P)$, so an interleaving of correct runs of $\mathcal{F}(P)$ and $\mathcal{F}(P')$ may owe it a liveness obligation that the run of $\mathcal{F}(P)$ does not owe. We exclude it.

▶ **Definition 21** (Contained). *A set of transactions R over a local-states function S is **contained** if for every $P \subset \Pi$, every $t = c \to c'$ in R with participants $Q \subseteq P$ and $c \in S(P)^Q$ is in $R(P)$.*

For a set of volition-guarded transactions, a volition-guarded transaction is taken by its machine transaction and a volition-guarded if-then-else transaction $s \to s'\,;\, s''$ guarded by p by its two branches, with source s.

▶ **Definition 22** (Transactions-Based Protocol). *Let S be a local-states function and R a set of machine transactions over S, in which distinct transactions have disjoint P-closures for every $P \subset \Pi$, with transaction equivalence $\sim$ (Definition 2). The **transactions-based protocol over** R, S, **and** $\sim$ assigns to each $P \subset \Pi$ the multiagent transition system $\mathcal{F}(P) := (S(P)^P, \{s_0\}^P, R(P)\!\uparrow\! P, {\sim}\!\uparrow\! P)$, where $R(P)\!\uparrow\! P$ and ${\sim}\!\uparrow\! P$ are the P-closures (Definition 11) of $R(P)$ and of $\sim$ restricted to $R(P)$. In particular $\mathcal{A}(P) = S(P)$, monotone as Definition 14 requires, $C(P) = S(P)^P$, and $c_0(P) = \{s_0\}^P$.*

By Definition 22, the transitions of $\mathcal{F}(P \cup P')$ are induced by the machine transactions in $R(P \cup P')$. Since liveness ranges over the equivalence classes of $\mathcal{F}(P \cup P')$ (Definition 8), any class $[t]$ in $R(P \cup P')/\!\sim$ whose transactions have participants spanning both P and P' could obstruct obliviousness if enabled in an interleaving; so could a class whose transactions have participants in P and that is not a class of $\mathcal{F}(P)$, which containment excludes. The following proposition identifies the conditions under which neither occurs.

▶ **Proposition 23.** *A transactions-based protocol over a contained set of transactions is oblivious provided that for every disjoint nonempty $P, P' \subset \Pi$, no equivalence class whose transactions have participants spanning both P and P' is ever enabled in any interleaving of correct runs of $\mathcal{F}(P)$ and $\mathcal{F}(P')$.*

3.3 Volitional Transactions-Based Grassroots Protocols

A volitional transactions-based protocol assigns to each set of agents the volitional multiagent transition system induced by its volition-guarded transactions. Obliviousness follows from a syntactic guard condition, in three steps. First, the Volitional Containment Lemma (Lemma 25) establishes that an agent's volitional state, in any configuration of $\mathcal{F}(P)$, never contains classes of transactions outside $R(P)$ (Definition 20). Second, Proposition 27 uses this invariant to reduce obliviousness to a condition on equivalence classes in the interleaving. Third, Corollary 28 establishes that condition whenever the set of transactions is contained and every volition-guarded transaction of two or more participants has a nonempty guard. A protocol so oblivious is grassroots once it is interactive (Definition 17). We then define the volitional notion: volitionally grassroots requires obliviousness, interactivity of groups each holding an agent that has not left, and the first interaction between any two groups willed by a member of each.

Throughout this subsection, $\sim$ is an equivalence on T_R, the machine transactions underlying the whole set R; for $T' \subseteq T_R$ we write $T'/\!\sim$ for the set of $\sim$-classes with a representative in T', and $[t]$ for the class of t in $T_R/\!\sim$. Hence $P \subseteq P'$ implies $T_{R(P)}/\!\sim \subseteq T_{R(P')}/\!\sim$.

▶ **Definition 24** (Volitional Transactions-Based Protocol)**.** *Let S be a local-states function, R a set of volition-guarded transactions over S with equivalence $\sim$, and I a set of inputs over S. The* ***protocol $\mathcal{F}$ over R, I, S, and $\sim$*** *assigns to each set of agents $P \subset \Pi$ the volitional multiagent transition system $\mathcal{F}(P)$ induced by $(S(P), R(P), I(P), \sim)$ over P (Definition 12), with $I(P)$ per Definition 20. In particular, $C(P) = \mathcal{A}(P)^P$ with agent state space $\mathcal{A}(P) := 2^{T_{R(P)}/\sim} \times S(P)$—monotone in P, as Definition 14 requires—and $c_0(P)$ has p-component $(\emptyset, s_0)$ for every $p \in P$.*

An input is unary, so an input transition is never an interaction, and it is in no class, so it incurs no liveness obligation; containment is asked of R and not of I: an input of p naming an agent q it does not hold is in $I(P \cup \{q\})$ and not in $I(P)$, and is owed by no run.

The obliviousness results below rest on the following invariant: in a group's own system an agent wills only transactions internal to the group, so the guard of a transaction that reaches another group cannot will it while the group runs alone, and the transaction is never enabled in an interleaving.

▶ **Lemma 25** (Volitional Containment)**.** *Let $\mathcal{F}$ be a volitional transactions-based protocol over a set of volition-guarded transactions R with equivalence $\sim$. For every $P \subset \Pi$, every configuration c of $\mathcal{F}(P)$, and every $p \in P$: $c_p^v \subseteq T_{R(P)}/\!\sim$, where $T_{R(P)} := \{t : (t, Q') \in R(P)$ for some $Q'\}$.*

Proof. By Definition 12, every configuration of $\mathcal{F}(P)$ is in $\mathcal{A}(P)^P$ with $\mathcal{A}(P) = 2^{T_{R(P)}/\sim} \times S(P)$, so $c_p^v \subseteq T_{R(P)}/\!\sim$ for every $p \in P$. ◀

▶ **Lemma 26** (Interleaving Safety)**.** *Let $\mathcal{F}$ be a volitional transactions-based protocol and $P, P' \subset \Pi$ disjoint and nonempty. Every finite prefix of an interleaving of a correct run of $\mathcal{F}(P)$ and a correct run of $\mathcal{F}(P')$ is a finite safe run of $\mathcal{F}(P \cup P')$.*

▶ **Proposition 27.** *A volitional transactions-based protocol over a contained set of transactions is oblivious provided that for every disjoint nonempty $P, P' \subset \Pi$, no equivalence class whose transactions have participants spanning both P and P' is ever enabled in any interleaving of correct runs of $\mathcal{F}(P)$ and $\mathcal{F}(P')$.*

When every volition-guarded transaction of two or more participants has a nonempty guard, the hypothesis of Proposition 27 is established by a uniform argument about volitions, without invoking platform-specific machine preconditions.

▶ **Corollary 28** (Guarded Obliviousness). *A volitional transactions-based protocol over a contained set of volition-guarded transactions R is oblivious if every volition-guarded transaction $(t, Q') \in R$ whose machine transaction t has two or more participants has a nonempty guard, $Q' \neq \emptyset$.*

Corollary 28 reduces obliviousness—for volitional transactions-based protocols—to containment and a syntactic check on guards. The two platforms of Section 4 apply it uniformly.

Containment is needed. Let $S(P) := 2^P$, and let an unguarded unary transaction of p add to its state an agent it does not hold, the additions of q by p forming one class. A correct run of $\mathcal{F}(\{p\})$ ends with p holding $\{p\}$, and one of $\mathcal{F}(\{q\})$ with q holding $\{q\}$. Their interleaving is not a correct run of $\mathcal{F}(\{p, q\})$, in which p adding q is enabled from some point on and never taken. Every transaction is unary, so the guard condition holds; the set is not contained, since p adding q has its source in $S(\{p\})$ and is not in $R(\{p\})$. Guarding the additions by their agent does not help where all additions by p form one class: p wills the class again after adding itself, and in $\mathcal{F}(\{p\})$ the class is then never enabled.

An agent has left when no transaction it guards can again be machine-enabled, so that its will never again has effect. Leaving is a machine condition: a platform records a person's death or incapacity, and the retirement of an AI agent operating a machine, by a machine state from which no transaction the agent guards is machine-enabled. We define leaving, and require interactivity only of groups each holding an agent that has not left.

▶ **Definition 29** (Left). *Let $\mathcal{F}$ be a volitional transactions-based protocol over R, $Q \subset \Pi$, c a configuration of $\mathcal{F}(Q)$, and $x \in Q$. Then x **has left** in c if no $(t, Q') \in R(Q)$ with $x \in Q'$ is machine-enabled in any configuration c' of $\mathcal{F}(Q)$ with $c \xrightarrow{*} c' \subseteq T(Q)$.*

▶ **Definition 30** (Interactive Among the Agents That Have Not Left). *A volitional transactions-based protocol $\mathcal{F}$ is **interactive among the agents that have not left** if for every $Q \subset \Pi$ and disjoint nonempty $P, P' \subseteq Q$, every prefix of a correct run of $\mathcal{F}(Q)$ containing no interaction between P and P', in whose last configuration some agent of P and some agent of P' have not left, is a prefix of a correct run of $\mathcal{F}(Q)$ that contains one.*

An interactive protocol is interactive among the agents that have not left, the latter requiring a continuation of fewer prefixes. A group all of whose agents have left can never be connected by an act one of them wills, and the definition asks nothing of it.

In a run of $\mathcal{F}(P \cup P')$ a member may will a transaction reaching the other group, which a group's own run forbids; the volitional notion constrains the resulting coupling:

▶ **Definition 31** (Volitionally Grassroots). *A volitional transactions-based protocol $\mathcal{F}$ is **volitionally grassroots** if it is oblivious, interactive among the agents that have not left, and, for every disjoint nonempty $P, P' \subset \Pi$ and every safe run of $\mathcal{F}(P \cup P')$ interactive between P and P', the first interaction of P and P' is induced by a volition-guarded transaction (t, Q') with $Q' \cap P \neq \emptyset$ and $Q' \cap P' \neq \emptyset$.*

Two groups thus become connected only by an act a person of each wills, and a grassroots protocol meeting the condition on first interactions is volitionally grassroots. An else transaction (Definition 6) is over a single participant and so is never an interaction; a binary transaction with an else-branch would let an agent of another group, who never wills it, trigger it, and could be a first interaction willed by neither, which is why Definition 5 admits only unary ones.

▶ **Corollary 32** (Volitional Coalescence). *Let $\mathcal{F}$ be a volitionally grassroots protocol, $Q \subset \Pi$, and $P, P' \subseteq Q$ disjoint instances in a prefix r of a correct run of $\mathcal{F}(Q)$, each holding an agent that has not left in the last configuration of r. Then r is a prefix of a correct run of $\mathcal{F}(Q)$ in which P and P' coalesce.*

Proof. As P and P' are distinct components of the interaction graph of r, no transition of r is an interaction between them, and each holds an agent that has not left in the last configuration of r; by Definition 30, r is a prefix of a correct run of $\mathcal{F}(Q)$ some transition of which is an interaction between an agent of P and one of P', so P and P' lie in one component of the interaction graph of that run. ◀

4 Grassroots Platforms via Volition-Guarded Transactions

We now present two grassroots platforms: grassroots social networks and grassroots currencies. For each platform, we specify its volition-guarded transactions, define the induced volitional multiagent transition system, prove platform-specific invariants, and prove the platform is grassroots via Corollary 28 and Definition 17. By *invariants* we mean properties preserved across all runs of the induced multiagent transition system; these are the platform-specific analogue of safety in the classical sense. In both platforms every transaction is guarded, so the condition of Definition 12 on the closures of unguarded transactions holds. Neither platform has inputs: a person names another by willing a transaction with them. Proofs are in Appendix B.

4.1 Grassroots Social Networks via Befriending and Unfriending

In a grassroots social network [42], the social graph is stored distributively under the control of the people themselves, with each person storing the local neighbourhood pertaining to them, and no third party having access unless explicitly granted. The original definition [42] was via a unary multiagent transition system; here both actions are specified as binary transactions.

Each agent p maintains, as its local state, a finite set $c_p \subseteq P$ recording the friends of p; initially $c_p = \emptyset$. Befriending adds q to c_p and p to c_q; unfriending removes them. Communication functions of a social network can be added, under the restriction that communication occurs only among friends [42]. A liveness theorem can be proven for this design [42], stating that if a person p who follows a person q is connected to q via a chain of mutual friends, each of them correct and following q, then p will eventually receive every item on q's feed. The specification of the grassroots social graph is the foundation for grassroots social networks with feeds, groups, messaging, explored elsewhere [42, 43, 48].

▶ **Definition 33** (Grassroots Social Graph). *The **grassroots social graph** SG is the protocol over the befriend and unfriend volition-guarded transactions of the Introduction, with local-states function $S(P) := 2^P$ and equivalence $\sim$ that identifies all befriend transactions of the same pair $\{p, q\}$ with each other and all unfriend transactions of the same pair with each other, per Definition 24.*

Invariants.

▶ **Lemma 34** (Mutuality). *Given a safe run r of SG, a configuration $c \in r$, and agents $p, q \in P$, $q \in c_p \iff p \in c_q$.*

We note that each configuration c in a safe run r of SG induces a graph with agents as vertices and an edge $p \leftrightarrow q$ when $q \in c_p$ (equivalently, by Lemma 34, when $p \in c_q$), and that the graphs induced by two consecutive configurations in r differ by exactly one added or removed edge.

Transaction equivalence. All befriend transactions of the same pair $\{p, q\}$—differing in the configurations in which they occur—form an equivalence class. Similarly for unfriend.

Liveness. Liveness applies to the befriend and unfriend classes (Definition 8). Unfriending is guarded by either p or q: once either person wills the class, the transaction becomes enabled and must eventually be taken. Befriending is guarded by both p and q: it becomes enabled only when both persons will the class, and must then eventually be taken.

Grassroots.

▶ **Corollary 35.** *The grassroots social graph SG is volitionally grassroots.*

4.2 Grassroots Currencies

Grassroots coins [28, 44] are units of debt that can be issued and traded digitally by any person. Each person's coins are backed by the goods and services they offer, priced in their own currency, with liquidity arising from mutual credit via coin exchange among persons that know and trust each other. Grassroots bonds [44] extend grassroots coins with a maturity date, so that a coin can be lent for an interest-bearing bond; digital social contracts—voluntary agreements among persons, specified, fulfilled, and enforced digitally—can then express the full gamut of financial instruments as the voluntary swap of bonds, including credit lines, loans, sale of debt, forward contracts, options, and escrow-based instruments [44]. Here we specify the coins.

▶ **Definition 36** (Coins, Lots)**.** *A **p-coin**, denoted $\mathbb{c}_p$, is a unit of debt issued by $p \in \Pi$. We let $\mathcal{B}(P) = \{\mathbb{c}_p : p \in P\}$ denote the coins of agents $P \subset \Pi$. Each agent p maintains as its local state a multiset c_p of members of $\mathcal{B}(P)$, initially $\emptyset$; we write $\cup$ and $\setminus$ for multiset union and difference. A **lot** $\mathbb{c}_u^k$ is a multiset of $k \geq 1$ u-coins.*

The grassroots coins volition-guarded transactions are the following, for $p \neq q \in \Pi$; the state of a participant not named is unchanged:

1. **Mint**, over $\{p\}$: $c'_p := c_p \cup \mathbb{c}_p^k$, $k \geq 1$. Guarded by p.
2. **Swap**, over $\{p, q\}$: $c'_p := (c_p \cup y) \setminus x$, $c'_q := (c_q \cup x) \setminus y$, provided $x \subseteq c_p$ and $y \subseteq c_q$ are lots. Guarded by $\{p, q\}$.
3. **Pay**, over $\{p, q\}$: $c'_p := c_p \setminus \mathbb{c}_q^k$, $c'_q := c_q \cup \mathbb{c}_q^k$, provided $\mathbb{c}_q^k \subseteq c_p$. Guarded by p.
4. **Redeem**, over $\{p, q\}$: $c'_p := (c_p \cup \{\mathbb{c}_u\}) \setminus \{\mathbb{c}_q\}$, $c'_q := (c_q \cup \{\mathbb{c}_q\}) \setminus \{\mathbb{c}_u\}$, provided $\mathbb{c}_q \in c_p$ and $\mathbb{c}_u \in c_q$, a coin of any issuer u. Guarded by p.

Minting is guarded by the minter. A swap is guarded by both parties. Paying and redeeming are guarded by the initiator: each is an obligation of the counterparty under its coins, so its volition is not required. A swap in which each party gives the other coins of their own issue opens a *mutual credit line*. In redemption, the bearer of a q-coin takes for it any coin q holds, regardless of who issued it. The contract from which the platform is compiled (Appendix E) requires the parties to a swap, a payment and a redemption to know each other; knowing is specified with keys and identities, outside this example, and here a person names another by willing a transaction with them.

▶ **Definition 37** (Grassroots Coins). *The **grassroots coins** protocol GC is the protocol over the volition-guarded transactions above with local-states function mapping each $P \subset \Pi$ to the multisets of members of $\mathcal{B}(P)$, and equivalence $\sim$ the least equivalence on their machine transactions that identifies Mint transactions by the same agent of the same lot, Swap transactions between the same pair exchanging the same lots, Pay transactions from the same payer to the same payee of the same number of coins, and Redeem transactions by the same bearer from the same issuer taking a coin of the same issuer, per Definition 24.*

One machine transaction can be several of these transactions, which are then in one class, and the guards of any one of them suffice to take it: a redemption is also the swap of the same two coins, so the bearer suffices; and the exchange of p's q-coin for q's p-coin is a redemption by either, so either suffices.

Invariants.

▶ **Lemma 38** (Conservation of Money). *In any safe run r of GC and any configuration $c \in r$, the number of p-coins held by all agents in c is the number of p-coins p has minted in the prefix of r ending in c.*

Liveness. Mint is guarded by the minting agent; it becomes enabled when the agent wills the class, and must then eventually be taken. Pay and Redeem are guarded by the initiator, and a redemption both ways by either party; once enabled, they must eventually be taken. Swap is guarded by both participants; it becomes enabled only when both persons will the class, and must then eventually be taken.

Grassroots.

▶ **Corollary 39.** *Grassroots coins are volitionally grassroots.*

5 Related Work

Atomic transactions. This work extends the notion of multiagent atomic transactions of [45] with volitions. Atomic transactions were investigated early in distributed computing, mostly in the context of database systems [26, 31, 29]. Most research since then has focused on their efficient and robust implementation [10, 12]. The integration of atomic transactions in programming languages has also been explored [9]. In terms of formal models of concurrency, the extension of CCS with atomic transactions has been investigated in the past [2, 15, 16], but seemingly without follow-on research. While transition systems have been the bedrock of abstract models of computation since the Turing machine, we are not aware of previous attempts to explore atomic transactions within their context.

Formal models of persons in concurrent systems. The formal methods tradition has a long lineage of modelling human agents as sources of nondeterminism alongside deterministic machines, but—to the best of our knowledge—without decomposing an agent into person and machine as components of its state. A detailed survey—covering Turing's choice machines, CSP and CCS, I/O automata and reactive systems, angelic/demonic nondeterminism, game structures and alternating-time temporal logic, ceremony analysis, and normative multiagent systems and electronic institutions—appears in Appendix F.

Across all traditions, the person is modelled as environment, opponent, error source, or unconstrained nondeterministic process—but always as an entity *external* to the agent, never as a formal component of the agent's state. The present work, to the best of our knowledge, is the first to decompose each agent's state into a machine state and a volitional state within

a multiagent transition system, with machine transactions conditioned on the volitions of the agents' persons, and with liveness arising from the interplay of personal volitions and machine obligations rather than from a designated set of live transitions.

6 Conclusion and Future Work

We have presented volition-guarded multiagent atomic transactions—a formal foundation for describing systems consisting of people operating machines—in which each agent's state decomposes into a machine state and a volitional state, and machine transactions are guarded by their people's volitions. We have developed the mathematical machinery needed to express the safety and liveness of grassroots platforms thus specified, and demonstrated the framework on two grassroots platforms: social networks and currencies. We have provided a simpler definition of grassroots that better captures the informal notion and excludes systems based on shared global data structures.

The framework extends naturally to platforms with richer guard structures—such as grassroots federations, in which transactions are guarded by supermajorities or assemblies of community members rather than by all participants—deferred to future work.

The original framework [39] also considered faulty computations and fault-tolerant implementations. We have adapted its notions of implementation and fault-resilient implementation to the present setting; establishing fault-tolerant implementations of the specifications presented here is left to future work.

References

1 Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000.

2 Lucia Acciai, Michele Boreale, and Silvano Dal Zilio. A concurrent calculus with atomic transactions. In *European Symposium on Programming*, pages 48–63. Springer, 2007.

3 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

4 Lorenzo Alvisi, Allen Clement, Alessandro Epasto, Silvio Lattanzi, and Alessandro Panconesi. Sok: The evolution of sybil defense via social networks. In *Proceedings of the 2013 IEEE symposium on security and privacy*, pages 382–396. IEEE, 2013.

5 Alexander Artikis, Marek Sergot, and Jeremy Pitt. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic*, 10(1):1–42, 2009.

6 Ralph-Johan Back and Joakim von Wright. *Refinement Calculus: A Systematic Introduction*. Springer, 1998.

7 Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

8 Matthew L. Bolton, Ellen J. Bass, and Radu I. Siminiceanu. Using formal verification to evaluate human-automation interaction: A review. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(3):488–503, 2013.

9 Johannes Borgström, Karthikeyan Bhargavan, and Andrew D Gordon. A compositional theory for stm haskell. In *Proceedings of the 2nd ACM SIGPLAN Symposium on Haskell*, pages 69–80, 2009.

10 Manuel Bravo and Alexey Gotsman. Reconfigurable atomic transaction commit. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 399–408, 2019.

11 Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 46–52, 2009. doi:10.1145/1578002.1578010.

12 Gregory Chockler and Alexey Gotsman. Multi-shot distributed transaction commit. *Distributed Computing*, 34:301–318, 2021.

13 Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Constructing scalable overlays for pub-sub with many topics. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 109–118, 2007.

14 Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 14–25, 2007.

15 Edsko de Vries, Vasileios Koutavas, and Matthew Hennessy. Communicating transactions. In *International Conference on Concurrency Theory*, pages 569–583. Springer, 2010.

16 Edsko De Vries, Vasileios Koutavas, and Matthew Hennessy. Liveness of communicating transactions. In *Asian Symposium on Programming Languages and Systems*, pages 392–407. Springer, 2010.

17 John R Douceur. The sybil attack. In *Proceedings of the international workshop on peer-to-peer systems*, pages 251–260, EU, 2002. Springer.

18 Carl Ellison. Ceremony design and analysis. Technical Report 2007/399, IACR, 2007.

19 Marc Esteva, Juan A. Rodríguez-Aguilar, Carles Sierra, Pere Garcia, and Josep Lluís Arcos. On the formal specification of electronic institutions. In *Agent-Mediated Electronic Commerce (AMEC)*, volume 1991 of *LNCS*, pages 126–147. Springer, 2001.

20 Daniel Halpern, Ariel D Procaccia, Ehud Shapiro, and Nimrod Talmon. Federated assemblies. *Proc AAAI 2025; arXiv preprint arXiv:2405.19129*, 2024. URL: `https://arxiv.org/abs/2405.19129`.

21 David Harel and Amir Pnueli. On the development of reactive systems. In *Logics and Models of Concurrent Systems*, volume 13 of *NATO ASI Series*, pages 477–498. Springer, 1985.

22 C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

23 Idit Keidar, Andrew Lewis-Pye, and Ehud Shapiro. Constitutional consensus. *arXiv preprint arXiv:2505.19216*, 2025. URL: `https://arxiv.org/abs/2505.19216`.

24 Anne-Marie Kermarrec, Erick Lavoie, and Christian Tschudin. Gossiping with append-only logs in secure-scuttlebutt. In *Proceedings of the 1st international workshop on distributed infrastructure for common good*, pages 19–24, 2020. `doi:10.1145/3428662.3428794`.

25 Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. Module checking. *Information and Computation*, 164(2):322–344, 2001.

26 Butler W Lampson. Chapter 11. atomic transactions. In *Distributed Systems—Architecture and Implementation: an Advanced Course*, pages 246–265. Springer, 1981.

27 Brian Neil Levine, Clay Shields, and N Boris Margolin. A survey of solutions to the sybil attack. *University of Massachusetts Amherst*, 7:224, 2006.

28 Andrew Lewis-Pye, Oded Naor, and Ehud Shapiro. Grassroots flash: A payment system for grassroots cryptocurrencies. *arXiv preprint arXiv:2309.13191*, 2023. URL: `https://arxiv.org/abs/2309.13191`.

29 Nancy Lynch, Michael Merritt, William Weihl, and Alan Fekete. A theory of atomic transactions. In *ICDT'88: 2nd International Conference on Database Theory Bruges, Belgium, August 31–September 2, 1988 Proceedings 2*, pages 41–71. Springer, 1988.

30 Nancy Lynch, Roberto Segala, and Frits Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, 2003.

31 Nancy A Lynch and Michael Merritt. *Atomic transactions: in concurrent and distributed systems*. Morgan Kaufmann Publishers Inc., 1993.

32 Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. Technical Report MIT/LCS/TM-373, MIT Laboratory for Computer Science, 1989.

33 Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems: First International Workshop, IPTPS 2002*, pages 53–65, Cambridge, MA, USA, March 2002. Springer. `doi:10.1007/3-540-45748-8\_5`.

34 Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.

35 Robin Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

36 Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system, 2008.

37 Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. Challenges in the decentralised web: The mastodon case. In *Proceedings of the internet measurement conference*, pages 217–229, 2019. doi:10.1145/3355369.3355572.

38 Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Genuine personal identifiers and mutual sureties for sybil-resilient community growth. In *International Conference on Social Informatics*, pages 320–332, EU, 2020. Springer, Springer. doi:10.1007/978-3-030-60975-7_24.

39 Ehud Shapiro. Multiagent transition systems for composing fault-resilient protocol stacks. *arXiv preprint arXiv:2112.13650*, 2021. URL: https://arxiv.org/abs/2112.13650.

40 Ehud Shapiro. Grassroots currencies: Turning mutual trust into liquidity. *arXiv preprint arXiv:2202.05619*, 2022. URL: https://arxiv.org/abs/2202.05619.

41 Ehud Shapiro. Grassroots distributed systems: Concept, examples, implementation and applications (brief announcement). In *37th International Symposium on Distributed Computing (DISC 2023). (Extended version: arXiv:2301.04391)*, pages 47:1–47:7, Italy, 2023. LIPICS. URL: https://arxiv.org/abs/2301.04391.

42 Ehud Shapiro. Grassroots social networking: Serverless, permissionless protocols for twitter/linkedin/whatsapp. In *OASIS '23*. Association for Computing Machinery, 2023. doi:10.1145/3599696.3612898.

43 Ehud Shapiro. Glp: A grassroots, multiagent, concurrent, logic programming language for ai (full version). *arXiv preprint arXiv:2510.15747, Summary in Proc. of ICLP'26, EPTCS 450, pp. 119–133*, 2025. URL: https://arxiv.org/abs/2510.15747.

44 Ehud Shapiro. Grassroots bonds: Financing by the people, for the people. arXiv:2603.13671, 2026. URL: https://arxiv.org/abs/2603.13671, arXiv:2603.13671.

45 Ehud Shapiro. Grassroots platforms with atomic transactions: Social graphs, cryptocurrencies, and democratic federations. In *Proceedings of the 27th International Conference on Distributed Computing and Networking*, pages 71–81, 2026. arXiv preprint arXiv:2502.11299. doi:10.1145/3772290.3772309.

46 Ehud Shapiro. Implementing grassroots logic programs with multiagent transition systems and ai (full version). *arXiv:2602.06934. Summary to appear in Proc. of LOPSTR+PPDP'26*, 2026. URL: https://arxiv.org/abs/2602.06934.

47 Ehud Shapiro. Types for grassroots logic programs (full version). *arXiv preprint arXiv:2601.17957*, 2026. URL: https://arxiv.org/abs/2601.17957.

48 Ehud Shapiro. Volition elicitation: Operational semantics for people and their machines. arXiv:2607.14138, 2026. URL: https://arxiv.org/abs/2607.14138, arXiv:2607.14138.

49 Nimrod Talmon and Ehud Shapiro. Grassroots federation: Fair democratic governance at scale. *arXiv preprint arXiv:2505.02208; also Proc. of AAMAS'26*, 2025. URL: https://arxiv.org/abs/2505.02208.

50 Dominic Tarr, Erick Lavoie, Aljoscha Meyer, and Christian Tschudin. Secure scuttlebutt: An identity-centric protocol for subjective and decentralized applications. In *Proceedings of the 6th ACM conference on information-centric networking*, pages 1–11, 2019. doi:10.1145/3357150.3357396.

51 TorrentFreak. BitTorrent Turns 20: The File-Sharing Revolution Revisited, 7 2021. Contains Bram Cohen's original statement: "BitTorrent's customer is etree. Etree is a loose-knit community of people who distribute live concert recordings online". URL: https://torrentfreak.com/bittorrent-turns-20-the-file-sharing-revolution-revisited-210702/.

52 Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1936.

53 Alan M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, s2-45(1):161–228, 1939.

A Further Examples of Grassroots Platforms Specified by Volition-Guarded Transactions

A grassroots platform is specified by a set of volition-guarded transactions over a local-states function; the formal machinery for deriving a transition system and proving it grassroots is introduced in Section 2. Here we present the volition-guarded transactions of a further grassroots platform, illustrating the range of guard structures that arise in practice; the coins and the social graph are specified in Section 4.

Child-safe social networks. A child-safe social network is a grassroots social network in which a child's partaking in online activities is subject to parental consent; its specification as a digital social contract is work in preparation, and its implementation in vGLP is reported in [48]. Each agent a maintains its friends $c_a \subseteq P$, the agents η_a it knows, whether it is a child $\chi_a \in \{0,1\}$, and the children $\kappa_a \subseteq P$ of whom it is a parent; an agent is a *child* when $\chi_a = 1$ and a *person* otherwise, and a *parent* of r when $r \in \kappa_a$. How an agent comes to know another, and a child to have a parent, is specified by the contract; components not named are unchanged.

1. **Child befriend**, over $\{r,s,p,q\}$, where $r \neq s$, $r \in \kappa_p$, $s \in \kappa_q$, $\chi_r = \chi_s = 1$, $\chi_p = \chi_q = 0$, $q \in c_p$ or $q = p$, $s \in \eta_r$, $r \in \eta_s$, and $s \notin c_r$: $c'_r := c_r \cup \{s\}$, $c'_s := c_s \cup \{r\}$. Guarded by $\{r,s,p,q\}$.
2. **Child unfriend**, over $\{r,s,p\}$, where $r \in \kappa_p$, $\chi_r = 1$, $\chi_p = 0$, $s \in c_r$ and $r \notin \kappa_s$: $c'_r := c_r \setminus \{s\}$, $c'_s := c_s \setminus \{r\}$. Guarded by p.

Two children who know each other become friends by an act of both and of a parent of each, the two parents being friends or one parent; the parents' states do not change. A parent of a child may end any friendship of the child but one with a parent of the child, and the two friends end it by Unfriend of the social graph.

Summary. The guard captures the essential distinction between voluntary and obligatory transactions. Befriending and voluntary swaps are guarded by all participants—they require mutual willingness. Unfriending, payments, redemptions and child unfriending are guarded by a single party—they are obligatory once initiated. Child befriending illustrates the most complex case: a quaternary guard requiring all four agents to be willing, two of them stationary. Unguarded transactions (guard $Q' = \emptyset$), such as group message delivery [48], are purely mechanical and require no volitions.

Sybils. Sybils [17] arise when a person operates two or more identities. Defending against sybils is a broad research area [27, 4], which is largely algorithmic, often evaluated experimentally. A formal treatment of sybils requires a formal notion of a person and of the correspondence between persons and the identifiers they own [38]. The present framework offers a natural vehicle for formally studying sybils and sybil resilience: since an agent is a person operating a machine, identified by the public key of the keypair its machine holds, a sybil is a single person operating two or more agents under distinct keypairs, and absent sybils agents are in one-to-one correspondence with persons. Using the current framework to formally study sybil resilience is a subject of future work.

B Deferred Proofs

A transaction and a transition are structurally identical—both are pairs of configurations—but differ in their role: a transaction is specified over its participants Q, the agents whose

states are preconditions for the transaction to occur, and says nothing about agents outside Q; different transactions may have different sets of participants. A transition, by contrast, is over a fixed set of agents P, as it is a building block of a transition system over P that consists of transitions over P. Given a set of transactions, each over its own set of participants, the closure operator induces from them a set of transitions over a fixed P, in which non-participants remain stationary.

A set of machine transactions R over S, each with participants $Q \subseteq P$, defines a multiagent transition system over S and P as follows:

▶ **Definition 40** (Transactions-Based Multiagent Transition System). *Given agents $P \subset \Pi$, states S with initial state $s_0 \in S$, and a set of transactions R, each $t \in R$ over some $Q \subseteq P$ and S, the **transactions-based multiagent transition system** over P, S, and R is the multiagent transition system $TS = (S^P, \{s_0\}^P, R{\uparrow}P, \emptyset)$, with $\sim$ the empty partial equivalence, so that no transition belongs to a class and no liveness obligation is imposed.*

In other words, one can fully specify a multiagent transition system over S and P simply by providing a set of transactions over S, each with participants $Q \subseteq P$.

Lemma 9 (Extension) (restated). *Every finite safe run of a transition system is a prefix of a correct run of it.*

Proof. Let $TS = (S, s_0, T, \sim)$ be a transition system and r a finite safe run of it. By Definition 8 the liveness classes $T/\sim$ are countably many; fix an enumeration $[t_1], [t_2], \ldots$ of them, finite or infinite. Extend r in stages, stage k taking $i = 1, \ldots, k$ in turn and, whenever $[t_i]$ is enabled at the current last state s, appending an s' with $s \to s' \in [t_i]$; the stages stop if at the start of one no class is enabled at the last state. Each stage appends finitely many states, so the construction yields a computation $\hat{r}$, safe by construction and having r as a prefix.

If the stages stop, $\hat{r}$ is finite and no class is enabled at its last state, so no class is enabled in every state of a suffix of $\hat{r}$, and $\hat{r}$ is live. Otherwise $\hat{r}$ is infinite; let $[t_m]$ be a class enabled in every state of a suffix of $\hat{r}$. Some stage $k_0 \geq m$ begins within that suffix, and at every stage $k \geq k_0$ the class $[t_m]$ is taken in turn and is enabled, so a member of $[t_m]$ occurs in the suffix. Hence $\hat{r}$ is live, and in both cases it is a correct run with r as a prefix. ◀

The following lemma records the compositionality of closure.

▶ **Lemma 41** (Compositionality of Closure). *For sets of machine transactions R, R' over S and any $P \subset \Pi$: $(R \cup R'){\uparrow}P = R{\uparrow}P \cup R'{\uparrow}P$, and $R \subseteq R'$ implies $R{\uparrow}P \subseteq R'{\uparrow}P$.*

Proof. Both are immediate from the definition $R{\uparrow}P := \bigcup_{t \in R} t{\uparrow}P$. ◀

Proposition 23 (restated). *A transactions-based protocol over a contained set of transactions is oblivious provided that for every disjoint nonempty $P, P' \subset \Pi$, no equivalence class whose transactions have participants spanning both P and P' is ever enabled in any interleaving of correct runs of $\mathcal{F}(P)$ and $\mathcal{F}(P')$.*

Proof. Let $\mathcal{F}$ be a transactions-based protocol over machine transactions R, local-states function S, and equivalence $\sim$. Let $P, P' \subset \Pi$ be disjoint and nonempty, $r = c_0, c_1, \ldots$ a correct run of $\mathcal{F}(P)$, $r' = d_0, d_1, \ldots$ a correct run of $\mathcal{F}(P')$, and $e = e_0, e_1, \ldots$ an interleaving of r and r'.

Safety. $e_0 = c_0(P \cup P')$ as noted above. Consider a P-step $e_k \to e_{k+1}$; the transition $c_{i_k} \to c_{i_{k+1}}$ of $\mathcal{F}(P)$ it lifts lies in $R(P){\uparrow}P$, hence in $t{\uparrow}P$ for some $t \in R(P)$ over participants

$Q \subseteq P$. In $e_k \to e_{k+1}$ the agents of Q do what they do in t (matching $c_{i_k} \to c_{i_{k+1}}$ on Q), the agents of $P \setminus Q$ are stationary, and the agents of P' are stationary because it is a P-step. Every agent outside Q is thus stationary and Q matches t, so $e_k \to e_{k+1} \in t{\uparrow}(P \cup P')$. Since $R(P) \subseteq R(P \cup P')$, $e_k \to e_{k+1} \in R(P \cup P'){\uparrow}(P \cup P')$, a transition of $\mathcal{F}(P \cup P')$. The case of a P'-step is symmetric.

Liveness. Suppose for contradiction that some class $[t] \in R(P \cup P')/\!\sim$ is enabled in some suffix of e with no member of $[t]$ taken in the suffix. By the hypothesis of the Proposition, every representative of $[t]$ enabled at any e_k has participants contained in P or contained in P'. Consider a representative t' enabled at some e_k in the suffix; without loss of generality its participants $Q \subseteq P$, and, $\sim$ being a transaction equivalence, every member of $[t]$ has participants Q. A member t'' of $[t]$ enabled at a configuration e_ℓ of the suffix has its source in the restriction of e_ℓ to Q, which is that of c_{i_ℓ}, in $S(P)^Q$; as R is contained, $t'' \in R(P)$, and t'' is enabled at c_{i_ℓ} in $\mathcal{F}(P)$. Since the interleaving exhausts r, the indices i_ℓ over the suffix cover a tail of r, so the class of $[t]$ in $\mathcal{F}(P)$ is enabled at every configuration of that tail; as no member of $[t]$ is taken in the suffix, none is taken along that tail of r. This contradicts correctness of r. The P' case is symmetric. ◀

Lemma 26 (Interleaving Safety) (restated). *Let $\mathcal{F}$ be a volitional transactions-based protocol and $P, P' \subset \Pi$ disjoint and nonempty. Every finite prefix of an interleaving of a correct run of $\mathcal{F}(P)$ and a correct run of $\mathcal{F}(P')$ is a finite safe run of $\mathcal{F}(P \cup P')$.*

Proof. Let $\mathcal{F}$ be over volition-guarded transactions R with equivalence $\sim$ and inputs I, $r = c_0, c_1, \ldots$ a correct run of $\mathcal{F}(P)$, $r' = d_0, d_1, \ldots$ a correct run of $\mathcal{F}(P')$, and $e = e_0, e_1, \ldots$ an interleaving of r and r'. Then $e_0 = c_0(P \cup P')$ as noted above. A P-step $e_k \to e_{k+1}$ lifts a transition $c_{i_k} \to c_{i_{k+1}}$ of $\mathcal{F}(P)$ that, by Definition 12, is a volition change of some $p \in P$, a volitional machine transaction induced by some $(t, Q') \in R(P)$, an else transaction induced by some volition-guarded if-then-else transaction in $R(P)$, or an input transition induced by some input in $I(P)$. A volition change of $p \in P$ is a volition change of $p \in P \cup P'$ in $\mathcal{F}(P \cup P')$, lifted with P'-agents unchanged because it is a P-step. A volitional machine transaction induced by $(t, Q') \in R(P)$, with $t = d \to d'$ over $Q \subseteq P$, is induced by $(t, Q') \in R(P \cup P')$: its machine and volitional preconditions (Definition 4(2)) hold at e_k because they hold at c_{i_k}, and P'-agents are machine-stationary because it is a P-step. The postcondition ${e_{k+1}}_p^v = {e_k}_p^v \setminus \{[t]\}$ holds on P-agents by the $\mathcal{F}(P)$-step; on each $p' \in P'$, by Lemma 25 applied to r', ${e_k}_{p'}^v = (d_{j_k})_{p'}^v \subseteq T_{R(P')}/\!\sim$, and $t \notin R(P')$ since $Q \subseteq P$ and $Q \cap P' = \emptyset$, so $[t] \notin {e_k}_{p'}^v$ and the postcondition holds vacuously. An else transaction induced by a volition-guarded if-then-else transaction $s \to s'\,;\, s''$ in $R(P)$ guarded by $p \in P$ is induced by the same transaction in $R(P \cup P')$ (Definition 6): its preconditions ${e_k}_p^m = s$ and $[s \to s'] \notin {e_k}_p^v$ hold at e_k because they hold at c_{i_k}, every agent other than p is machine-stationary, and no volitional state changes. An input transition of $p \in P$ induced by an input in $I(P)$ is induced by the same input in $I(P \cup P')$ (Definition 7): its precondition ${e_k}_p^m = s$ holds at e_k because it holds at c_{i_k}, every agent other than p is machine-stationary, and no volitional state changes. Hence $e_k \to e_{k+1}$ is a transition of $\mathcal{F}(P \cup P')$. The case of a P'-step is symmetric. ◀

Proposition 27 (restated). *A volitional transactions-based protocol over a contained set of transactions is oblivious provided that for every disjoint nonempty $P, P' \subset \Pi$, no equivalence class whose transactions have participants spanning both P and P' is ever enabled in any interleaving of correct runs of $\mathcal{F}(P)$ and $\mathcal{F}(P')$.*

Proof. Let $P, P' \subset \Pi$ be disjoint and nonempty and e an interleaving of a correct run of $\mathcal{F}(P)$ and a correct run of $\mathcal{F}(P')$. It is safe, every finite prefix of it being a finite safe run of

$\mathcal{F}(P \cup P')$ by Lemma 26, and live by the Liveness argument of Proposition 23, applied to the enabled representatives of a class: a volition-guarded transaction whose guards hold its class at e_ℓ, as they do at c_{i_ℓ}, or an else-branch whose guard does not hold the class of its then-branch at e_ℓ, as at c_{i_ℓ}; by containment either is in $R(P)$ and enabled at c_{i_ℓ} in $\mathcal{F}(P)$. ◀

Corollary 28 (Guarded Obliviousness) (restated). *A volitional transactions-based protocol over a contained set of volition-guarded transactions R is oblivious if every volition-guarded transaction $(t, Q') \in R$ whose machine transaction t has two or more participants has a nonempty guard, $Q' \neq \emptyset$.*

Proof. We verify the hypothesis of Proposition 27. Let $P, P' \subset \Pi$ be disjoint and nonempty, e an interleaving of a correct run of $\mathcal{F}(P)$ and a correct run of $\mathcal{F}(P')$, $[t]$ an equivalence class of $R(P \cup P')/\sim$ whose participants Q span both P and P', and (t, Q') a representative. Since Q has participants in both groups, $|Q| \geq 2$, so by hypothesis $Q' \neq \emptyset$; pick $q \in Q'$ and, without loss of generality, $q \in P$. Since Q has a participant in P', $t \notin R(P)$ by Definition 20; and by well-formedness of $\sim$ (Definition 2), every $t' \sim t$ has the same participants as t, so $[t] \notin T_{R(P)}/\sim$. By Lemma 25 applied to the P-run, $e_{k_q}^v \subseteq T_{R(P)}/\sim$ at every e_k, so $[t] \notin e_{k_q}^v$ and the guard on q fails. Hence (t, Q') is not enabled at any e_k. ◀

Lemma 34 (Mutuality) (restated). *Given a safe run r of SG, a configuration $c \in r$, and agents $p, q \in P$, $q \in c_p \iff p \in c_q$.*

Proof. By induction on the length of the run $r = c_0, c_1, \ldots, c_n$. In the initial configuration $c_0 = \emptyset^P$ the biconditional holds vacuously. Assume the lemma holds for c_n, and consider the transition $c_n \to c_{n+1}$. If it is a volition change, all machine states are unchanged and the claim is immediate. Otherwise it is a Befriend or Unfriend for some pair $\{p, q\}$; both modify c_p and c_q symmetrically—Befriend adds q to c_p and p to c_q, Unfriend removes them—preserving the biconditional. For all other pairs $\{r, s\}$, the local states are unchanged. ◀

Corollary 35 (restated). *The grassroots social graph SG is volitionally grassroots.*

Proof. *Grassroots:* The befriend and unfriend transactions are contained (Definition 21): each changes the friend sets of its two participants by adding or removing each other. Befriend and unfriend are guarded—befriend by $\{p, q\}$, unfriend by p or by q—so every volition-guarded transaction of two or more participants has a nonempty guard, and by Corollary 28, SG is oblivious. For $Q \subset \Pi$ and disjoint nonempty $P, P' \subseteq Q$, take $p \in P$ and $q \in P'$ and let $\hat{r}$ be a prefix of a correct run of $SG(Q)$ containing no interaction between P and P'. A befriend of p and q is an interaction between P and P' and is the only transaction that puts q in c_p, so $q \notin c_p$ in the last configuration of $\hat{r}$, $p \notin c_q$ there by Lemma 34, and befriend(p, q) is enabled there once willed. Extend $\hat{r}$ by a volition change of p adding [befriend(p, q)], one of q adding it, and the befriend, which changes the local states of both p and q and takes c_p to $c_p \cup \{q\} \notin S(\{p\})$, and is therefore an interaction between p and q and hence between P and P'. The result is a finite safe run of $SG(Q)$ having $\hat{r}$ as a prefix and containing an interaction between P and P', and by Lemma 9 it is a prefix of a correct run of $SG(Q)$. Thus SG is oblivious and interactive, hence grassroots by Definition 17, and interactive among the agents that have not left (Definition 30).

Volitionally grassroots: By Definition 16, the first interaction of disjoint nonempty P, P' in any safe interactive run changes the state of an agent in each group; in SG it is therefore a befriend or an unfriend between an agent of P and one of P'. An unfriend of $p \in P$ and $q \in P'$ requires $q \in c_p$, an existing friendship that only an earlier befriend of p and

q—itself an interaction—can create; so the first interaction is not an unfriend. It is therefore a befriend, guarded by both its participants, one in P and one in P'. Hence SG is volitionally grassroots. ◀

Lemma 38 (Conservation of Money) (restated). *In any safe run r of GC and any configuration $c \in r$, the number of p-coins held by all agents in c is the number of p-coins p has minted in the prefix of r ending in c.*

Proof. Mint adds new p-coins at p. Swap, Pay and Redeem move coins between their two participants without creating or destroying any, and volition changes leave all machine states unchanged. Hence the number of p-coins held by all agents is the number p has minted. ◀

Corollary 39 (restated). *Grassroots coins are volitionally grassroots.*

Proof. *Grassroots:* The transactions of GC are contained (Definition 21): each puts into a participant's holdings coins of a participant or coins a participant held. Mint, Pay and Redeem are guarded by p and Swap by both participants, so every volition-guarded transaction of two or more participants has a nonempty guard, and by Corollary 28, GC is oblivious. For $Q \subset \Pi$ and disjoint nonempty $P, P' \subseteq Q$, take $p \in P$ and $q \in P'$ and let $\hat{r}$ be a prefix of a correct run of $GC(Q)$ containing no interaction between P and P'. Extend $\hat{r}$ by p willing and taking a Mint of p-coins, q willing and taking a Mint of q-coins, and p and q both willing a Swap of those lots and taking it; the swap changes the local states of both p and q and puts q-coins in p's holdings, which no local state of $GC(\{p\})$ carries, and is therefore an interaction between p and q and hence between P and P'. The result is a finite safe run of $GC(Q)$ having $\hat{r}$ as a prefix and containing an interaction between P and P', and by Lemma 9 it is a prefix of a correct run of $GC(Q)$. Thus GC is oblivious and interactive, hence grassroots by Definition 17, and interactive among the agents that have not left (Definition 30).

Volitionally grassroots: By Definition 16, the first interaction of disjoint nonempty P, P' in any safe interactive run changes the state of an agent in each group; Mint changes the state of one agent, so in GC the first interaction is a Swap, Pay or Redeem between an agent of P and one of P'. A Pay or Redeem in which $p \in P$ acts towards $q \in P'$ requires p to hold a q-coin. By the proof of Conservation of Money (Lemma 38), every q-coin originates in a Mint by q, and every other transaction moves coins between its participants; so the first transaction that puts a q-coin at an agent $x \in P$ moves it from an agent of P', changes the states of both, and puts at x a coin no local state of $GC(\{x\})$ carries—itself an interaction. So the first interaction is neither a Pay nor a Redeem. It is therefore a Swap, guarded by both its participants, one in P and one in P'. Hence GC is volitionally grassroots. ◀

Relation to the original grassroots definition. The original definition of grassroots protocols [41, 45] used a subset relation ($P \subset P'$) and conditions on transition availability, without incorporating liveness. The new definition uses disjoint groups and interleavings of correct runs. The two definitions are not comparable in general—neither implies the other—but we informally claim that, for the platforms considered here, both definitions agree.

C Systems Based on Shared Global Data Structures Are Not Grassroots

Federated systems such as Mastodon [37] are oblivious, as servers in one group can ignore servers in the other, but are not interactive: a group of clients P without a server cannot do

more when joined by another group of clients P', also without a server. Any protocol that employs a shared global data structure—whether replicated (Blockchain [36]) or distributed (DHT [33], IPFS [7])—is not oblivious, and hence not grassroots; the argument covers trackerless BitTorrent [51], which relies on a DHT. Below we prove that Bitcoin is not grassroots and show that the same argument applies to distributed hash tables and IPFS.

We model Bitcoin as a protocol (Definition 14) in which the local state of an agent is a set of blocks, initially the genesis block, and a set of agents it knows, initially the other members of a set $B \subset \Pi$ of *bootnodes* for a member of B and empty otherwise; its transitions are the closures of two machine transactions: **Mine**, over $\{p\}$, which adds to p's blocks a block named by p that p does not hold; and **Propagate**, over $\{p, q\}$ with p known to q, which adds to q's blocks a block p holds and q does not. The Propagates of one block to one agent form one class and a Mine is in no class, so in a correct run every block held by an agent is eventually held by every agent that knows it.

▶ **Proposition 42.** *Bitcoin is not grassroots.*

Proof. We argue that Bitcoin is not oblivious. Consider disjoint $P, P' \subset \Pi$ each containing a bootnode, $q \in P$ and $q' \in P'$, a correct run of $\mathcal{F}(P)$ in which q mines a block b, a correct run of $\mathcal{F}(P')$, and an interleaving e of the two. In e, q' knows q from the initial configuration, and from the step at which q mines b on, q holds b and q' does not, since every step of e is a P-step or a P'-step and the Propagate of b to q' is over $\{q, q'\}$. So the class of the Propagates of b to q' is enabled in every configuration of a suffix of e and no member of it is taken; e is not live, hence not a correct run of $\mathcal{F}(P \cup P')$, and Bitcoin is not oblivious. ◀

Distributed Hash Tables and IPFS. The same argument applies to systems based on distributed hash tables [33] and distributed file systems such as IPFS [7]. In a DHT, a lookup by a member of P may require routing through a member of P', so that a lookup transition that succeeds when P runs alone does not correspond to a valid transition in the combined system where routing tables reflect both groups. More precisely: in an interleaving of correct runs of P and P', each group builds its own routing table over its own members. A lookup in the combined system $P \cup P'$ must route through the combined table, but in the interleaving no routing entry connecting the two groups is ever established, so lookups that should succeed in $P \cup P'$ (by routing through P') instead fail or route incorrectly. The liveness condition requiring that lookups for keys stored by correct agents eventually succeed is thus violated. The argument for IPFS is analogous, as it relies on a DHT (specifically, a Kademlia-based DHT [33]) for content discovery. Trackerless BitTorrent [51] relies on a Kademlia-based DHT for peer discovery, so the argument applies to it as well.

D Implementations, Faults, and Resilience

An implementation relates a specification transition system to one that implements it, via a mapping of states. We adapt the notion of implementation among transition systems [39, 46], with safety and liveness as defined above.

▶ **Definition 43** (Implementation). *Given transition systems $TS = (S, s_0, T, \sim)$, the **specification**, and $TS' = (S', s'_0, T', \sim')$, an **implementation of** TS **by** TS' is a function $\sigma : S' \to S$ with $\sigma(s'_0) = s_0$, in which case the pair (TS', σ) is an **implementation of** TS. Given a computation $r' = s'_1, s'_2, \ldots$ of TS', $\sigma(r')$ is the computation $\sigma(s'_1), \sigma(s'_2), \ldots$ with consecutive repetitions removed; a transition $s' \to d'$ of TS' with $\sigma(s') = \sigma(d')$ is a **stutter**, contributing no step to $\sigma(r')$.*

▶ **Definition 44** (Correct and Complete Implementation)**.** *An implementation (TS', σ) of TS is **correct** if σ maps every correct run of TS' to a correct run of TS, and **complete** if every correct run r of TS is $\sigma(r')$ for some correct run r' of TS'.*

The original framework [39] also accounts for faulty behaviour and its resilience, which we adapt to the present setting.

▶ **Definition 45** (Safety Fault, Fault-Resilient Implementation)**.** *Let (TS', σ) be an implementation of TS, with $TS' = (S', s'_0, T', \sim')$. A **safety fault** is a set $F \subseteq S'^2 \setminus T'$ of **faulty transitions**, and a computation **performs** F if it includes a transition in F. The implementation is **F-resilient** if σ maps every live run $r' \subseteq T' \cup F$ of TS' to a correct run of TS.*

Faulty transitions lie outside T', hence in no class of $\sim'$, so they impose no liveness requirement.

The dual of a safety fault is a **liveness fault**: rather than acting incorrectly, correct agents cease to act, so that obligations that ought to be met are not. A safety fault enlarges the transitions that may occur; a liveness fault waives some of the obligations that liveness would otherwise impose.

▶ **Definition 46** (Liveness Fault, Liveness-Fault-Resilient Implementation)**.** *Let (TS', σ) be an implementation of TS, with $TS' = (S', s'_0, T', \sim')$. A **liveness fault** is a set $\Lambda \subseteq T'/\!\sim'$ of liveness classes. A run **performs** Λ if it is not live with respect to some class in Λ. A run is **Λ-live** if no class $[t] \in T'/\!\sim' \setminus \Lambda$ is enabled in every state of some suffix with no member of $[t]$ occurring in the suffix—that is, live for $\sim'$ with the classes in Λ removed from its domain. The implementation is **Λ-resilient** if σ maps every safe Λ-live run $r' \subseteq T'$ of TS' to a correct run of TS.*

With $\Lambda = \emptyset$ the Λ-live runs are exactly the live runs. In a volitional multiagent transition system the canonical liveness fault is **fail-stop**: when a set of agents $D \subseteq P$ cease to act, the waived classes are $\Lambda_D := \{[t] \in T'/\!\sim' : \text{some participant of } t \text{ is in } D\}$, every class some member of which requires a stopped agent. This is well-defined because $\sim'$-equivalent transactions share participants (Definition 2); and a class is waived whichever of its guards is live, since every member of it has a stopped participant.

The instance-level notion of resilient implementation that composes along a protocol stack is developed in [39] and can be adapted to this setting.

E Implementation

The specifications presented here provide the formal foundation for grassroots platforms. Here, we discuss how these platforms can be implemented, focusing on the role of GLP as the implementation language.

GLP as implementation language. Grassroots Logic Programs (GLP) [43] is designed for implementing grassroots platforms on networked smartphones. A multiagent implementation of GLP, madGLP, has been developed and proven to be a correct and outcome-complete implementation of maGLP, and the madGLP protocol has been proven grassroots [46].

The abstraction cascade. A grassroots platform is specified as a digital social contract, whose clauses are written as act schemas, work in preparation; a compiler reads the act schemas and compiles them to vGLP [48], to GLP, and to the app the people use. A contract also carries pragmas, which name the screens of the app and the acts its buttons propose,

and which do not affect the semantics of its acts. Volition-guarded transactions are a side branch of this cascade: the transactions of a contract are printed from its act schemas, and carry no pragma. The repository of GLP holds every contract, its act schemas, its pragmas and its transactions.

AI-derived implementations. The mathematical foundations presented here and in related papers have been used by AI to derive working implementations: (*i*) a workstation-based implementation of concurrent GLP and a smartphone-based multiagent implementation of GLP, both in Dart, derived from the formal operational semantics [46]; (*ii*) a GLP implementation of the grassroots social graph, child-safe social networking with parental-consent-based befriending and group membership [48]; (*iii*) a GLP implementation of grassroots bonds, including a running six-agent village market scenario exercising symmetric and asymmetric credit, payments, redemption, escrow, and sale of debt [44]; and (*iv*) a moded type system for GLP, implemented in Dart from a mathematical specification [47].

Enforcement. Enforcement of the digital social contract—ensuring that participants cannot deviate from the protocol as programmed—is achieved via mutual attestation among the participants' machines, as described in work on secure GLP.

F Formal Models of Persons in Concurrent Systems

The formal methods tradition has a long lineage of modelling human agents as sources of nondeterminism alongside deterministic machines, but—to the best of our knowledge—without decomposing an agent into person and machine as components of its state.

Turing's choice machines. Before defining what we now call Turing machines (automatic machines, or a-machines), Turing [52] introduced *choice machines* (c-machines), "whose motion is only partially determined by the configuration"—at designated states, the machine "cannot go on until some arbitrary choice has been made by an external operator." The external operator is a person who freely chooses between alternatives; the sequence of choices determines which computation unfolds. Turing immediately set c-machines aside, showing that any c-machine computation can be enumerated by an a-machine. His 1939 oracle machines [53] extend this further: the oracle "cannot be a machine" and provides answers the computation cannot derive internally. Both formalisms model the person as *external* to the machine, providing input at designated points.

Process algebras. Hoare's CSP [22] provides the cleanest process-algebraic encoding of human choice. External choice ($\Box$) offers the environment—potentially a person—a selection among initial events, while internal choice ($\sqcap$) is resolved by the system. However, the person remains *outside* the system boundary: CSP models what the person *does*, not what the person *is willing to do*. Milner's CCS [34] uses a single summation operator without formally separating internal from external nondeterminism at the syntactic level. In his later work on bigraphs [35], Milner makes the scope explicit: agents "can be artificial, as in computing systems... or they can be natural, e.g. communicating humans." Both CSP and CCS model agents uniformly—there is no formal distinction between a person and a machine within the same agent.

I/O automata and reactive systems. Lynch and Tuttle's I/O automata [32] partition actions into input (environment-controlled), output (automaton-controlled), and internal actions. The key property is *input-enabling*: an automaton cannot block input actions, so the environment—potentially a human operator—can act at any moment. The Hybrid I/O Automata extension [30] explicitly states that HIOAs are "intended to model all components

of hybrid systems, including... humans." The person, however, is part of the environment, not a component of the automaton's state. Harel and Pnueli's reactive systems paradigm [21] draws the foundational dichotomy between transformational systems (batch, terminating) and reactive systems (ongoing interaction with environment). The system is deterministic; all nondeterminism is attributed to the environment. This paradigm was explicitly motivated by human-machine interaction, yet the formalism treats the human as environment rather than as a component of the system.

Angelic and demonic nondeterminism. The distinction between angelic and demonic nondeterminism, developed by Back and von Wright [6] in the refinement calculus, provides a semantic treatment relevant to the person/machine boundary. Demonic nondeterminism models adversarial environments (the worst-case choice is made); angelic nondeterminism models cooperative choices (the best-case choice is made). This duality is the closest precursor to the volition/obligation distinction in the present work: a volitional transaction guarded by both parties (both must be willing) versus one guarded by either party (either can force it). Two distinctions separate the frameworks. First, the refinement calculus operates within a sequential program framework, not a multiagent transition system. Second, angelic nondeterminism is a point-of-choice semantics: a choice is resolved locally at each transition, with no residue carried forward. Volitions, in contrast, are persistent, inspectable state that accumulates across transitions; a guard condition reads an agent's record of willing over the history of the run, not a single local resolution. This shift—from choice as a point-semantic primitive to choice as state—is what lets volitions be shared, compared, and reasoned about within the transition system, rather than external to it.

Game structures and alternating-time temporal logic. Module checking [25] models open systems with the environment fully adversarial. Alternating-time temporal logic (ATL) [3] interprets formulas over concurrent game structures where multiple agents simultaneously choose actions. Game semantics [1] models computation as dialogue between Proponent (program, following a deterministic strategy) and Opponent (environment, making free moves). These frameworks treat agents as symmetric players but do not decompose a single agent into person and machine components.

Ceremony analysis and human-interactive verification. Ellison's ceremony analysis [18] extends security protocol analysis to include human participants as protocol nodes. Bolton's Enhanced Operator Function Model (EOFM) [8] translates hierarchical human task models into state machines for model checking, with the human as the sole source of nondeterminism. Both treat human nondeterminism as a source of error to be verified against, rather than as a source of legitimate volition to be formally recorded.

Normative multiagent systems and electronic institutions. Electronic institutions [19] model multiagent interaction as dialogical frameworks where human and software agents are treated uniformly as role-playing entities. Normative multiagent systems [5] use Event Calculus to specify societies where agents "may fail to, or even choose not to, conform to the specifications." These approaches model norms that constrain agents, but do not decompose an agent's state into machine and volitional components, nor do they formalise the distinction between transactions requiring all parties to be willing and those that are obligatory once initiated.