Volitional Agents and Grassroots Protocols

Structure

1) Review: transition systems
2) Multi-agent transition systems (MATS)
3) Volitional states, volitional transactions, Volitional MATS.
4) Grassroots protocols.
5) Volitional Transactions-based protocols.
6) Example: Child-safe networks.

Transition systems

A **transition system** (S, T, L) is specified by:

- A set of **states** S (with a special designated *initial state*).
- A set of correct **transitions** $T \subseteq S^2$. Each $t \in T$ is a pair (s, s'), where $s \neq s'$.
- A set L of disjoint subsets of T. Each element of L can be thought of as a liveness class...

Transition systems

A **transition system** (S, T, L) is specified by:

- A set of **states** S (with a special designated *initial state*).
- A set of correct **transitions** $T \subseteq S^2$. Each $t \in T$ is a pair (s, s'), where $s \neq s'$.
- A set L of disjoint subsets of T. Each element of L can be thought of as a liveness class...

We'll see how L is used to specify liveness in a bit...

Transition systems

A **transition system** (S, T, L) is specified by:

- A set of **states** S (with a special designated *initial state*).
- A set of correct **transitions** $T \subseteq S^2$. Each $t \in T$ is a pair (s, s'), where $s \neq s'$.
- A set L of disjoint subsets of T. Each element of L can be thought of as a liveness class...

A **run** is a sequence of states in S, beginning with the initial state. A run is **safe** if every consecutive pair of states is a correct transition in T.

A **transition system** (S, T, L) is specified by:

- A set of **states** S (with a special designated *initial state*).
- A set of correct **transitions** $T \subseteq S^2$. Each $t \in T$ is a pair (s, s'), where $s \neq s'$.
- A set L of disjoint subsets of T. Each element of L can be thought of as a liveness class...

A set of transitions $M \in L$ is **enabled** at $s \in S$ if there exists some s' with $(s, s') \in M$, i.e., if it is now possible to make a transition in M.

A run r is **live** if no $M \in L$ is enabled in every state of a suffix of r without any transition in M occurring in the suffix.

A **transition system** (S, T, L) is specified by:

- A set of **states** S (with a special designated *initial state*).
- A set of correct **transitions** $T \subseteq S^2$. Each $t \in T$ is a pair (s, s'), where $s \neq s'$.
- A set L of disjoint subsets of T. Each element of L can be thought of as a liveness class...

A set of transitions $M \in L$ is **enabled** at $s \in S$ if there exists some s' with $(s, s') \in M$, i.e., if it is now possible to make a transition in M.

A run r is **live** if no $M \in L$ is enabled in every state of a suffix of r without any transition in M occurring in the suffix.

A run is **correct** if it is safe and live.

Multiagent Transition Systems

Given a set of **agents** P and a set of *underlying* states S, a **configuration** is an element of S^P, i.e., a configuration assigns a state in S to each agent in P.

The states of the MATS are configurations, and T is now contains pairs of configurations.

An Aside

Today we won't focus on faulty behaviour: we are focussed on high level specifications that determine correct behaviour.

Resilience to faults can be studied using the notion of **implementations**:

- For transition systems $TS = (S, T, L)$ and $TS' = (S', T', L')$, an implementation of TS by TS' is a function mapping states of S' to states of S.

An Aside

Today we won't focus on faulty behaviour: we are focussed on high level specifications that determine correct behaviour.

Resilience to faults can be studied using the notion of **implementations**:

- For transition systems $TS = (S, T, L)$ and $TS' = (S', T', L')$, an implementation of TS by TS' is a function mapping states of S' to states of S.
- An implementation is *correct* if it maps correct runs of TS' to correct runs of TS.

An Aside

Today we won't focus on faulty behaviour: we are focussed on high level specifications that determine correct behaviour.

Resilience to faults can be studied using the notion of **implementations**:

- For transition systems $TS = (S, T, L)$ and $TS' = (S', T', L')$, an implementation of TS by TS' is a function mapping states of S' to states of S.
- An implementation is *correct* if it maps correct runs of TS' to correct runs of TS.
- It is *complete* if every correct run of TS is mapped to by some run of TS'.

An Aside

Today we won't focus on faulty behaviour: we are focussed on high level specifications that determine correct behaviour.

Resilience to faults can be studied using the notion of **implementations**:

- For transition systems $TS = (S, T, L)$ and $TS' = (S', T', L')$, an implementation of TS by TS' is a function mapping states of S' to states of S.
- An implementation is *correct* if it maps correct runs of TS' to correct runs of TS.
- It is *complete* if every correct run of TS is mapped to by some run of TS'.

Given a high level specification TS, one can then consider lower level implementations, together with notions of fault tolerance for those implementations.

Volitions

Next, we want to introduce Volitional MATS.

First we have to introduce volitional states, and certain types of volitional transitions...

Volitions

Next, we want to introduce **volitions**: these are not just choices made at a point in time, but are first class objects in the model, reflected in persistent state.

The basic idea is that we separate an agent's state into two parts: their **machine state** and their **volitions**...

Agent state

Machine state	Volitional state
$s \in S$	A set of transactions they are willing to engage in

Volitions

To formalise this, we distinguish between transitions and **transactions**: a transaction just specifies the change in state for some subset $Q \subseteq P$.

Volitions

To formalise this, we distinguish between transitions and **transactions**: a transaction just specifies the change in state for some subset $Q \subseteq P$.

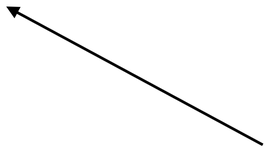
Guarded Machine Transactions. Given an arbitrary set S of *machine states* and agents $Q \subseteq P$, a **machine configuration** over Q is a member of S^Q, and a **machine transaction** over *participants* Q is a pair $(c, c') \in (S^Q)^2$ such that $c \neq c'$.

Volitions

To formalise this, we distinguish between transitions and **transactions**: a transaction just specifies the change in state for some subset $Q \subseteq P$.

Guarded Machine Transactions. Given an arbitrary set S of *machine states* and agents $Q \subseteq P$, a **machine configuration** over Q is a member of S^Q, and a **machine transaction** over *participants* Q is a pair $(c, c') \in (S^Q)^2$ such that $c \neq c'$.

Given such a machine transaction t, a **guarded transaction** over t is a pair (t, Q') where $Q' \subseteq Q$ are its **guards**.



The agents that must be willing for the transaction to take place.

Volitions

To formalise this, we distinguish between transitions and **transactions**: a transaction just specifies the change in state for some subset $Q \subseteq P$.

Guarded Machine Transactions. Given an arbitrary set S of *machine states* and agents $Q \subseteq P$, a **machine configuration** over Q is a member of S^Q, and a **machine transaction** over *participants* Q is a pair $(c, c') \in (S^Q)^2$ such that $c \neq c'$.

Given such a machine transaction t, a **guarded transaction** over t is a pair (t, Q') where $Q' \subseteq Q$ are its **guards**.

> 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 guarded transactions over the same machine transaction, $(t, \{p\})$ and $(t, \{q\})$, so that either person's volition suffices.

Volitions

Next, we want to formally define volitional states and correct volitional transitions (basically those where the volitional state of the guards is as required)...

...but there is a slight complication that we really have to consider equivalence classes of transactions here...

Volitions

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

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

Volitions

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

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

> For example, a running example in this talk will be Child-Safe Social Networks. Here, we consider an evolving network of contacts, and the machine state of an agent is its present contacts. In this context, befriending another agent might be thought of as a single transaction, but is really an equivalence class of transactions (one for each state before the befriending).

Volitions

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

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

Now we can formally define volitional states and correct volitional transitions...

Volitions

Given agents P, machine states S, a set of machine transactions T, 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**.
- An **agent configuration** c is a member $c \in \mathcal{A}^P$.

The volitional state is a set of (equivalence classes of) transactions the agent is willing to execute

Volitions

Given agents P, machine states S, a set of machine transactions T, 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**.
- An **agent configuration** c is a member $c \in \mathcal{A}^P$.

We let c_p^m denote the machine state of agent p in configuration c, while c_p^v denotes their volitional state in c.

Volitions

Volitional transactions are of two kinds:

- **Volition changes**: a single agent changes their volitional state.

Volitions

Volitional transactions are of two kinds:

- **Volition changes**: a single agent changes their volitional state.
- A **volitional machine transaction** induced by a guarded machine transaction (t, Q') for some $t = (d, d')$ with participants Q is a pair (c, c') where:
  - $c \neq c'$ are agent configurations such that $[t] \in c_q^v$ for every $q \in Q'$;

Every guard wills the transaction.

Volitions

Volitional transactions are of two kinds:

- **Volition changes**: a single agent changes their volitional state.
- A **volitional machine transaction** induced by a guarded machine transaction (t, Q') for some $t = (d, d')$ with participants Q is a pair (c, c') where:
  - $c \neq c'$ are agent configurations such that $[t] \in c^v_q$ for every $q \in Q'$;
  - $c^m_p = d_p$ and $c'^m_p = d'_p$ for every $p \in Q$, and;

The machine transaction is that specified by t.

Volitions

Volitional transactions are of two kinds:

- **Volition changes**: a single agent changes their volitional state.
- A **volitional machine transaction** induced by a guarded machine transaction (t, Q') for some $t = (d, d')$ with participants Q is a pair (c, c') where:
  - $c \neq c'$ are agent configurations such that $[t] \in c_q^v$ for every $q \in Q'$;
  - $c_p^m = d_p$ and $c'^m_p = d'_p$ for every $p \in Q$, and;
  - $c'^v_p = c_p^v \setminus \{[t]\}$ for every $p \in Q$.

Once the transaction is executed, we remove it from the volitional state.

Example

A **child-safe social network** is a grassroots social network in which a child's partaking in online activities is subject to parental consent.

Each agent p maintains a set of friends $c_p^m \subseteq P$ (their machine state). In what follows, r, s are children with respective parents p, q. We specify the protocol using two types of guarded machine transaction.

Example

A **child-safe social network** is a grassroots social network in which a child's partaking in online activities is subject to parental consent.

Each agent p maintains a set of friends $c_p^m \subseteq P$ (their machine state). In what follows, r, s are children with respective parents p, q. We specify the protocol using two types of guarded machine transaction.

(1) **Child befriend**: $(c')_r^m := c_r^m \cup \{s\}$, $(c')_s^m := c_s^m \cup \{r\}$, provided $q \in c_p^m$ and $s \notin c_r^m$. Guarded by $\{r, s, p, q\}$.
(2) **Child unfriend**: $(c')_r^m := c_r^m \setminus \{s\}$, $(c')_s^m := c_s^m \setminus \{r\}$, provided $s \in c_r^m$. Guarded by any one of $\{r, s, p, q\}$.

The precondition $q \in c_p^m$ requires the parents to be friends. Child befriending requires all four—both children and both parents—to be willing; child unfriending can be initiated by any one of the four.

Now that we know what states and transitions look like, we can define Volitional MATS...

Volitional Multiagent Transition Systems

Consider a set of agents P, machine states S, a set R of guarded machine transactions, and an equivalence $\sim$ on the set $T_R := \{t : (t, Q') \in R \text{ for some } Q'\}$ of underlying machine transactions.

The **volitional multiagent transition system induced by** $(S, R, \sim)$ **over** P is the multiagent transition system $(\mathcal{A}^P, T_V, \sim_V)$ where:

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

Volitional Multiagent Transition Systems

Consider a set of agents P, machine states S, a set R of guarded machine transactions, and an equivalence $\sim$ on the set $T_R := \{t : (t, Q') \in R \text{ for some } Q'\}$ of underlying machine transactions.

The **volitional multiagent transition system induced by** $(S, R, \sim)$ **over** P is the multiagent transition system $(\mathcal{A}^P, T_V, \sim_V)$ where:

(1) $\mathcal{A} := 2^{T_R/\sim} \times S$ is the **agent state space**;
(2) T_V consists of all transitions of one of two forms: (i) a **volition change**; or (ii) a **volitional machine transaction** induced by some guarded machine transaction $(t, Q') \in R$;

Volitional Multiagent Transition Systems

Consider a set of agents P, machine states S, a set R of guarded machine transactions, and an equivalence $\sim$ on the set $T_R := \{t : (t, Q') \in R \text{ for some } Q'\}$ of underlying machine transactions.

The **volitional multiagent transition system induced by** $(S, R, \sim)$ **over** P is the multiagent transition system $(\mathcal{A}^P, T_V, \sim_V)$ where:

(1) $\mathcal{A} := 2^{T_R/\sim} \times S$ is the **agent state space**;
(2) T_V consists of all transitions of one of two forms: (i) a **volition change**; or (ii) a **volitional machine transaction** induced by some guarded machine transaction $(t, Q') \in R$;
(3) $\sim_V$ is the set of equivalence classes of volitional machine transitions induced by R, relating two of them whenever their inducing machine transactions are $\sim$-equivalent.

The liveness requirement is that, for any volitional machine transaction enabled on all states in a suffix of a run, some equivalent transaction is eventually executed.

Structure

1) Review: transition systems
2) Multi-agent transition systems (MATS)
3) Volitional states, volitional transactions, Volitional MATS.
4) Grassroots protocols.
5) Volitional Transactions-based protocols.
6) Example: Child-safe networks.

Grassroots Protocols

We consider a potentially infinite set of agents Π: any protocol must run for any finite $P \subset \Pi$.

A **protocol** $\mathcal{F}$ is a family of multiagent transition systems that has exactly one multiagent transition system $\mathcal{F}(P) = (S(P), T(P), L(P))$ for every $P \subset \Pi$, where $P \subseteq P'$ implies $S(P) \subseteq S(P')$.

Grassroots Protocols

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...

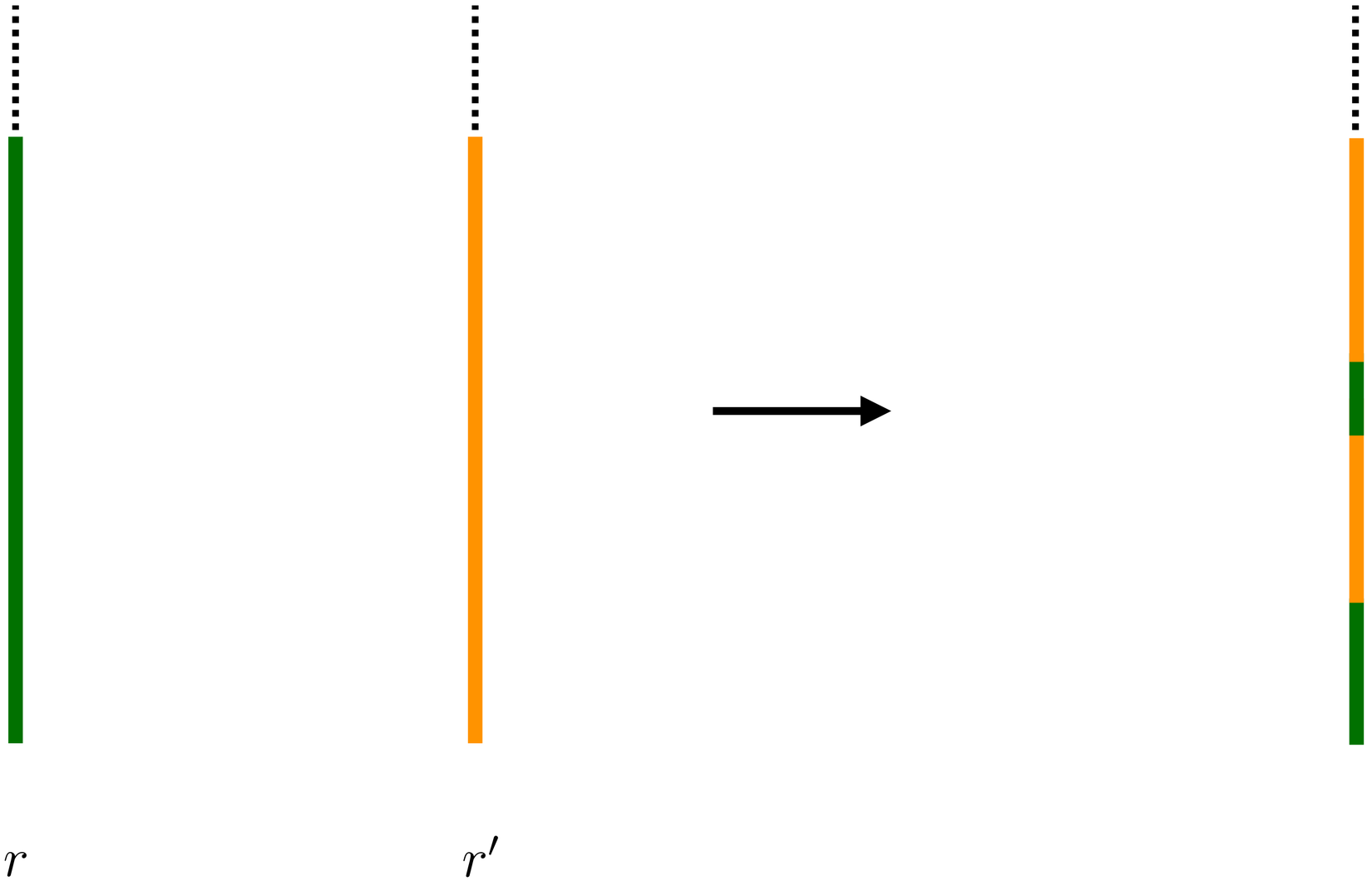
Grassroots Protocols

Let $P, P' \subset \Pi$ be disjoint nonempty sets of agents, r a run of $\mathcal{F}(P)$, and r' a run of $\mathcal{F}(P')$.

Roughly, an **interleaving** of r and r' is a sequence $c_0, c_1, \ldots$ of configurations in $C(P \cup P')$ such that each transition is either the *next* transition in r (leaving the states of members of P' unchanged) or the next transition in r' (leaving the states of members of P unchanged).

We also require that each transition in r is eventually executed, and similarly for r'.

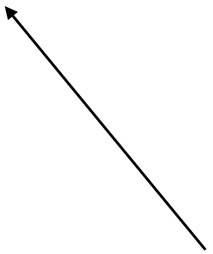
Grassroots Protocols



Interactive runs

Let $P, P' \subset \Pi$ be disjoint and nonempty. For a configuration c over $P \cup P'$ and $Q \subseteq P \cup P'$, write $c|_Q$ for the restriction of c to Q.

A transition (c, c') of $\mathcal{F}(P \cup P')$ is an **interaction between P and P'** if $c|_P \neq c'|_P$ and $c|_{P'} \neq c'|_{P'}$, and $(c|_P, c'|_P)$ is not a transition of $\mathcal{F}(P)$ or $(c|_{P'}, c'|_{P'})$ is not a transition of $\mathcal{F}(P')$.



Something genuinely new, that couldn't happen in $\mathcal{F}(P)$ or $\mathcal{F}(P')$.

Interactive runs

Let $P, P' \subset \Pi$ be disjoint and nonempty. For a configuration c over $P \cup P'$ and $Q \subseteq P \cup P'$, write $c|_Q$ for the restriction of c to Q.

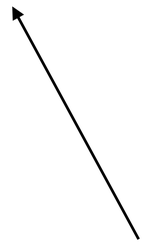
A transition (c, c') of $\mathcal{F}(P \cup P')$ is an **interaction between P and P'** if $c|_P \neq c'|_P$ and $c|_{P'} \neq c'|_{P'}$, and $(c|_P, c'|_P)$ is not a transition of $\mathcal{F}(P)$ or $(c|_{P'}, c'|_{P'})$ is not a transition of $\mathcal{F}(P')$.

A run r of $\mathcal{F}(P \cup P')$ is **interactive between P and P'** if some transition of r is an interaction between P and P'.

Grassroots Protocols

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 $\mathcal{F}(P \cup P')$.

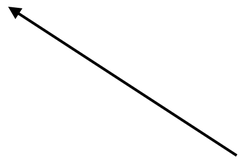


P and P' don't have to interact in an interesting way.

Grassroots Protocols

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 disjoint nonempty $P, P' \subset \Pi$, some correct run of $\mathcal{F}(P \cup P')$ is interactive between P and P'.



P and P' can interact in an interesting way.

Grassroots Protocols

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 disjoint nonempty $P, P' \subset \Pi$, some correct run of $\mathcal{F}(P \cup P')$ is interactive between P and P'.
(3) **grassroots** if it is oblivious and interactive.

Volitional Transactions-Based Protocols

Let S be a function mapping any P to a set of machine states. Let R be a set of guarded transactions over S with equivalence $\sim$.

Volitional Transactions-Based Protocols

Let S be a function mapping any P to a set of machine states. Let R be a set of guarded transactions over S with equivalence $\sim$.

The volitional transactions-based protocol $\mathcal{F}$ over R, S, and $\sim$ assigns to each set of agents $P \subset \Pi$ the volitional multiagent transition system $\mathcal{F}(P)$ induced by $(S(P), R(P), \sim)$ over P. In particular, $C(P) = \mathcal{A}(P)^P$ with agent state space $\mathcal{A}(P) := 2^{T_{R(P)}/\sim} \times S(P)$.

basically, the correct volitional machine transactions are those induced by guarded transactions in R

Volitional Transactions-Based Protocols

One of the reasons volitional transactions-based protocols are nice to work with is the following result...

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

Back to child-safe networks...

Recall that each agent p maintains a set of friends $c_p^m \subseteq P$ (their machine state). In what follows, r, s are children with respective parents p, q. We specify the protocol using two types of guarded machine transaction.

(1) **Child befriend**: $(c')_r^m := c_r^m \cup \{s\}$, $(c')_s^m := c_s^m \cup \{r\}$, provided $q \in c_p^m$ and $s \notin c_r^m$. Guarded by $\{r, s, p, q\}$.
(2) **Child unfriend**: $(c')_r^m := c_r^m \setminus \{s\}$, $(c')_s^m := c_s^m \setminus \{r\}$, provided $s \in c_r^m$. Guarded by any one of $\{r, s, p, q\}$.

Back to child-safe networks...

Recall that each agent p maintains a set of friends $c_p^m \subseteq P$ (their machine state). In what follows, r, s are children with respective parents p, q. We specify the protocol using two types of guarded machine transaction.

(1) **Child befriend**: $(c')_r^m := c_r^m \cup \{s\}$, $(c')_s^m := c_s^m \cup \{r\}$, provided $q \in c_p^m$ and $s \notin c_r^m$. Guarded by $\{r, s, p, q\}$.
(2) **Child unfriend**: $(c')_r^m := c_r^m \setminus \{s\}$, $(c')_s^m := c_s^m \setminus \{r\}$, provided $s \in c_r^m$. Guarded by any one of $\{r, s, p, q\}$.

To prove that the protocol is grassroots, we need to prove it is oblivious and interactive. Proving that it is interactive is striaghtforward: any run of $\mathcal{F}(P \cup P')$ in which a member of P befriends a member of P' is interactive.

But given the previous theorem, obliviousness also follows directly.