## Keywords
consensus, games semantics, concurrency, message-passing, types, Curry-Howard

## ABSTRACT
We present an axiomatic framework for reasoning about consensus protocols.

# Consensus Games

L.G. Meredith
Biosimilarity, LLC
lgreg.meredith@biosimilarity.com

Vlad Zamfir
Ethereum
vldzmfr@gmail.com

Vitalik Buterin
Ethereum
vitalik.buterin@ethereum.org

Matthew Wampler-Doty
MIT
matthew.wampler.doty@gmail.com

Aaron Fisher
Colony
aron.mathsguy@gmail.com

## 1. INTRODUCTION AND MOTIVATION

Our aim is to describe an axiomatic framework for analyzing and comparing a wide range of consensus protocols. The intended outcome is a proof of the safety and termination properties of class of consensus protocols, under a wide class of fault and network conditions.

### 1.1 Related work

This work builds on several strands of research. One of the main strands is the line of investigation begun by Linear Types Can Change The Blockchain (pdf, lex, hangout video) . Another strand is the seminal work by Vlad Zamfir and Vitalik Buterin on Casper. Another strand of research is the work on games semantics for programs initiated by Abramsky, Hyland, and Ong.

### 1.2 Outline of paper

First we set out the basic elements and intuitions of the technical framework. Then we build up a compositional model of typed consensus protocols in which types describe safety and liveness properties.

## 2. TECHNICAL FRAMEWORK

Consensus games take place amongst a set, P, of players who are concurrently sending messages to each other. Each message is an announcement of a position in the game, which we dub here a bet, indicating a player's belief in a claim that the players will agree on a value v at the conclusion of the game. Bets are described by the following recursive domain equation.

### 2.1 Bets

DEFINITION 2.1.1 (BET).

$$B[P, C, F] = 1 + P \times P \times C \times F \times \mathsf{List}[B[P, C, F]]$$

Recall the equation for n-ary $A$-labelled trees is given by

$$\mathsf{Tree}[A] = 1 + A \times \mathsf{List}[Tree[A]]$$

and note that $B[P, C, F] = \mathsf{Tree}[P \times P \times C \times F]$

In the sequel we will let $b$, $b'$, etc range over bets and for convenience, we give logical accessors to components of the bet via the projections, $\pi_i$. Following executable specification languages like Haskell or Scala, we can improve readability

$$\begin{aligned}
\mathsf{src}(\mathsf{inr}((s, t, c, f, j)) &= s \\
\mathsf{trgt}(\mathsf{inr}((s, t, c, f, j)) &= t \\
\mathsf{claim}(\mathsf{inr}((s, t, c, f, j)) &= c \\
\mathsf{belief}(\mathsf{inr}((s, t, c, f, j)) &= f \\
\mathsf{justify}(\mathsf{inr}((s, t, c, f, j)) &= j
\end{aligned}$$

We understand these components of the data structure to have the following meaning

- $\mathsf{src}(b)$ represents the source or origin of the bet, i.e. the player who is sending the bet;
- $\mathsf{trgt}(b)$ represents the target or destination of the bet message, i.e. the player who is the intended recipient of the bet;
- $\mathsf{claim}(b)$ represents the claim of the bet, i.e. the source player's estimate of a value;
- $\mathsf{belief}(b)$ represents the source player's confidence in the claim given the evidence in the justification;

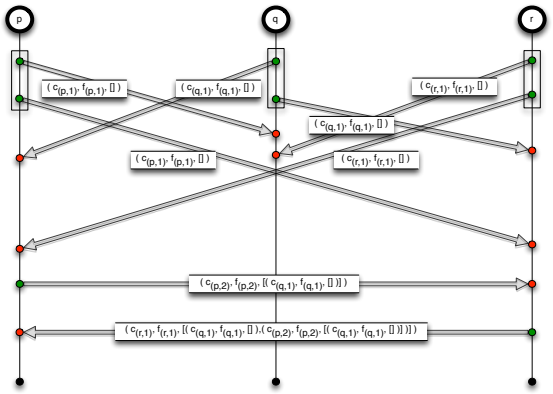


Figure 1: Non-redundant bet

- $\mathsf{justify}(b)$ represents the justification or evidence of the claim or estimate made in the bet

### 2.1.1 Temporal order

Notice that there is an implicit description of time in the bet data structure in the sense that the justification of a bet contains the history of observations leading up to that bet; and this can be formalized by

Definition 2.1.2 (Temporal order).

$$b_1 \sqsupseteq b_2 \iff b_2 \in \mathsf{justify}(b_1) or \exists b_3 \in \mathsf{justify}(b_1).b_3 \sqsupseteq b_2$$

From this perspective the choice of whether the collection of bets in the justification of a particular bet, say $b$, is a ordered (List) as in definition 2.1.1 or unordered (Set) correlates to whether history is totally ordered or partially ordered. When the justification is ordered, the temporal order relation can by made total using the order of the justification together with the ordering of the tree. When interpreting bets as messages in a message-passing protocol, the choice of an unordered justification can be used to represent message arrival order non-determinism.

### 2.1.2 Visualizing bets

The data structure allows to recover a great deal of information about network communication. To see this, let's consider the following example.

$$\begin{aligned}
b_{p\to q/1} &= inr((p, q, c_{(p,1)}, f_{(p,1)}, [])) \\
b_{p\to r/1} &= inr((p, r, c_{(p,1)}, f_{(p,1)}, [])) \\
b_{q\to p/1} &= inr((q, p, c_{(q,1)}, f_{(q,1)}, [])) \\
b_{q\to r/1} &= inr((q, r, c_{(q,1)}, f_{(q,1)}, [])) \\
b_{r\to p/1} &= inr((r, p, c_{(r,1)}, f_{(r,1)}, [])) \\
b_{r\to q/1} &= inr((r, q, c_{(r,1)}, f_{(r,1)}, [])) \\
b_{p\to r/2} &= inr((p, r, c_{(p,2)}, f_{(p,2)}, [b_{q\to p/1}])) \\
b_{r\to p/2} &= inr((p, r, c_{(p,2)}, f_{(p,2)}, [b_{q\to r/1}, b_{p\to r/2}]))
\end{aligned}$$

This collection of bets can be visualized as an *interaction diagram* amongst the players, $p, q$, and $r$ as depicted in figure 1. Time runs from top to bottom. Arrows indicate the communication of bets, with source and target information depicted by both the arrow orientation and the green and red vertices. The latter provides a visual aid to readily identify the view a particular player has of the network at any given time. Specifically, a player's view at a given time is simply the collection of red vertices up to the depth corresponding to that time. In a somewhat non-standard way, we use rectangular boxes around vertices to indicate concurrency, or no ordering on particular communication events. Thus, for example, $b_{p\to q/1}$ and $b_{p\to r/1}$ are indicated as sent in parallel. It is only $q$'s view of events that determines their arrival order for $q$.

### 2.1.3 Information density

The specification for bets allows for a wide range of structures that are not necessarily ideal message structures. Consider the following bet extending the example above.

$$b_{p\to r/3} = inr((p, r, c_{(p,3)}, f_{(p,3)}, [b_{q\to p/1}, b_{r\to p/2}]))$$

Because $b_{r\to p/2}$ already includes $b_{q\to p/1}$, indicating that $r$ has already seen $p$'s evidence for what it has seen from $q$, and that $p$ was the provider of said evidence, including this evidence is clearly redundant. While redundancy can be beneficial in certain contexts, it is generally harmful here, providing an overhead on communication that can be used maliciously. So, we want to restrict our attention to redundancy-free bets. Notice, however, that the notion is somewhat subtle. For example, even though $b_{p\to r/2}$ includes $b_{q\to r/1}$, the justification of $b_{r\to p/2}$ cannot be considered redundant because it provides potentially useful information about the arrival order of bets at $r$.

We can formalize the notion of redundancy-free bets using a notion of path. Since a bet is really just a labelled n-ary tree, we can talk about paths in a bet,

which are simply sequences of bets, i.e. elements of $\mathsf{List}[B[P,C,F]]$. We denote the set of paths from $b$ with $\mathsf{paths}(b)$ and use $\sigma$ to range over $\mathsf{paths}(b)$, $|\sigma|$ to denote the length of $\sigma$ and 0-based indexing notation to denote positions in the path. Thus, for example, $\sigma(0)$ is the initial bet in the path, and $\sigma(|\sigma|-1)$ denotes the final bet in the path.

DEFINITION 2.1.3 (REDUNDANCY-FREE). *A bet, b, is redundancy-free iff*

$$b' \in \mathsf{justify}(b) \Rightarrow \forall \sigma \in List[B[P,C,F]]. \; \sigma(0) \in \mathsf{justify}(b) \;\&\; src(\sigma(|\sigma|-1)) = src(b) \Rightarrow b \notin \mathsf{justify}(\sigma(|\sigma|-1))$$

## 2.2 Strategies

A consensus game is determined, in part, by a move relation, $\vdash$, on pairs of sets of bets, $(M,N) \in \mathcal{P}(B[P,C,F]) \times \mathcal{P}(B[P,C,F])$ indicating the legal moves. The statement $M \vdash N$ indicates that given the set of bets $M$, a player who plays by the rules produces a bet in $N$.

DEFINITION 2.2.1 (STRATEGY). *A strategy, s, is a map $s : \mathcal{P}(B[P,C,F]) \to B[P,C,F]$, that produces a new bet from the information contained in a set of bets given as arguments to s. We use S to range over sets of strategies, that is, $S \subseteq \mathcal{P}(B[P,C,F]) \to B[P,C,F]$.*

DEFINITION 2.2.2. *A strategy, s, is said to be legal w.r.t $\vdash$, alternatively, $\vdash$-legal, iff*

$$M \subseteq B[P,C,F] \longrightarrow \exists N \subseteq B[P,C,F].s(M) \in N \& M \vdash N.$$

In the sequel we will want to associate players with strategies, and use $S$ to range over the set of partial maps $P \Rightarrow (\mathcal{P}(B[P,C,F]) \Rightarrow B[P,C,F]+1)$.

## 2.3 Relationship to traditional notions of consensus

Roughly speaking, a consensus protocol implies a legal move relation. N.b. illegal moves are evidence of Byzantine behavior. Traditionally, a consensus protocol also implies a strategy, and Byzantine behaviour in the canonical consensus literature includes playing any strategy that isnâĂŹt âĂIJprotocolâĂİ. The framework of definitions above is more relaxed. There will be legal strategies that would be considered Byzantine in traditional consensus literature. For example, a Byzantine node, in the classical consensus literature, could delay publishing bets without making any illegal moves.

## 2.4 Games

We capture the dynamics of game play through the notion of a history of a game.

DEFINITION 2.4.1. *Given a move relation and a set of players, P, we can describe a complete game history H as a partial map*

$$H : P \times \mathcal{L}\rangle\int\sqcup[\mathcal{S}]\sqcup[B[P,C,F]] + \mathcal{S}]\sqcup[B[P,C,F]]] \to \mathcal{S}]\sqcup[B[P,C,F]] +$$

- $H(p,l) \mapsto \rangle\backslash\nabla(A')$ *indicates that $A'$ is the last set of bets p will see in this history, while*
- $H(p,l) \mapsto \rangle\backslash\updownarrow(A')$ *indicates that p will receive bets after $A'$*
- $H(p,l) \mapsto \perp$ *indicates that H is undefined, i.e. that p will never see l and so cannot see an extension of l.*

DEFINITION 2.4.2. *We define the H-messages of l for p from $p'$, $msgH(l,p,p)$, as*

$$msgH(l,p,p') = trgt(H(p',l),p)$$

This is under ideal network conditions where all messages sent arrive. It will be convenient to address network conditions separately from the choices of players to send messages. For this we will adopt a notation $msgH(l,p,p;C)$, which we detail in a separate section.

## 2.5 Composition of games

# 3. FORMULAE

# 4. SAFETY AND LIVENESS

# 5. CONCLUSIONS AND FUTURE WORK

TBD

*Acknowledgments.* TBD