Keywords
π-calculus, proof-of-stake, blockchain, types, Curry-Howard

ABSTRACT
We present Casper, a proof-of-stake protocol, and it's formal specification in π-calculus.

Formally introducing Casper

Vlad Zamfir
Ethereum
vldzmfr@gmail.com

L.G. Meredith
CSO, Synereo
greg@synereo.com

1. INTRODUCTION AND MOTIVATION

1.0.1 Related work

1.0.2 Organization of the rest of the paper

2. CASPER, INFORMALLY

3. THE CALCULUS

One notable feature of the π-calculus is its ability to succinctly and faithfully model a number of phenomena of concurrent and distributed computing. Competition for resources amongst autonomously executing processes is a case in point. The expression

$$x?(y) \Rightarrow P \mid x!(u) \mid x?(v) \Rightarrow Q$$

is made by composing three processes, two of which, $x?(y) \Rightarrow P$ and $x?(v) \Rightarrow Q$ are seeking input from channel x before they launch their respective continuations, P and/or Q; while the third, $x!(u)$, is supplying output on that same said channel. Only one of the input-guarded processes will win, receiving u and binding it to the input variable, y, or respectively, v in the body of the corresponding continuation – while the loser remains in the input-guarded state awaiting input along channel x. The calculus is equinanimous, treating both outcomes as equally likely, and in this regard is unlike its sequential counterpart, the λ-calculus, in that it is not *confluent*. There is no guarantee that the different branches of computation must eventually converge. Note that just adding a new-scope around the expression

$$(\mathsf{new}\ x)(x?(y) \Rightarrow P \mid x!(u) \mid x?(v) \Rightarrow Q)$$

ensures that the competition is for a local resource, hidden from any external observer.

3.1 Our running process calculus

3.1.1 Syntax

$P ::= 0$	stopped process
$\mid x?(y_1, \ldots, y_n) \Rightarrow P$	input
$\mid x!(y_1, \ldots, y_n)$	output
$\mid (\mathsf{new}\ x)P$	new channel
$\mid P \mid Q$	parallel

Due to space limitations we do not treat replication, $!P$.

3.1.2 Free and bound names

$$\begin{aligned}
&\mathcal{FN}(0) := \emptyset \\
&\mathcal{FN}(x?(y_1, \ldots, y_n) \Rightarrow P) := \\
&\quad \{x\} \cup (\mathcal{FN}(P) \setminus \{y_1, \ldots y_n\}) \\
&\mathcal{FN}(x!(y_1, \ldots, y_n)) := \{x, y_1, \ldots, y_n\} \\
&\mathcal{FN}((\mathsf{new}\ x)P) := \mathcal{FN}(P) \setminus \{x\} \\
&\mathcal{FN}(P \mid Q) := \mathcal{FN}(P) \cup \mathcal{FN}(Q)
\end{aligned}$$

An occurrence of x in a process P is *bound* if it is not free. The set of names occurring in a process (bound or free) is denoted by $\mathcal{N}(P)$.

3.1.3 Structural congruence

The *structural congruence* of processes, noted $\equiv$, is the least congruence containing α-equivalence, $\equiv_\alpha$, making $(P, |, 0)$ into commutative monoids and satisfying

$$(\mathsf{new}\ x)(\mathsf{new}\ x)P \equiv (\mathsf{new}\ x)P$$

$$(\mathsf{new}\ x)(\mathsf{new}\ y)P \equiv (\mathsf{new}\ y)(\mathsf{new}\ x)P$$

$$((\mathsf{new}\ x)P) \mid Q \equiv (\mathsf{new}\ x)(P \mid Q)$$

3.1.4 Operational Semantics

$$\frac{|\vec{y}| = |\vec{z}|}{x?(\vec{y}) \Rightarrow P \mid x!(\vec{z}) \to P\{\vec{z}/\vec{y}\}} \quad (\textsc{Comm})$$

In addition, we have the following context rules:

$$\frac{P \to P'}{P \mid Q \to P' \mid Q} \qquad (\textsc{Par})$$

$$\frac{P \to P'}{(\mathsf{new}\ x)P \to (\mathsf{new}\ x)P'} \qquad (\textsc{New})$$

$$\frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q} \qquad (\textsc{Equiv})$$

3.1.5 Bisimulation

Definition 3.1.1. *An* observation relation, $\downarrow$ *is the smallest relation satisfying the rules below.*

$$\frac{}{x!(\vec{y}) \downarrow x} \qquad (\textsc{Out-barb})$$

$$\frac{P \downarrow x \ \textit{or}\ Q \downarrow x}{P \mid Q \downarrow x} \qquad (\textsc{Par-barb})$$

Notice that $x?(y) \Rightarrow P$ has no barb. Indeed, in π-calculus as well as other asynchronous calculi, an observer has no direct means to detect if a sent message has been received or not.

Definition 3.1.2. *An* barbed bisimulation, *is a symmetric binary relation $\mathcal{S}$ between agents such that $P\ \mathcal{S}\ Q$ implies:*

1. *If $P \to P'$ then $Q \to Q'$ and $P'\ \mathcal{S}\ Q'$.*
2. *If $P \downarrow x$, then $Q \downarrow x$.*

P is barbed bisimilar to Q, written $P \dot{\approx} Q$, if $P\ \mathcal{S}\ Q$ for some barbed bisimulation $\mathcal{S}$.

4. FORMALLY INTRODUCING CASPER

Acknowledgments. We would like to acknowledge Anthony D'Onofrio for putting us in touch.

5. APPENDIX: BECAUSE EVERY TECHNICAL PAPER NEEDS AN APPENDIX