

Applying the EPR verification methodology to Casper

Giuliano Losa

January 24, 2025

Contents

This is an experiment dating from 2017 in which I shortened an Isabelle/HOL proof by Yoichi Hirai from about 1000 lines to less than 100 by encoding things in EPR as much as possible.

It seems that this proof was later re-used by Yoichi Hirai and then translated to Coq by Runtime Verification:

<https://github.com/runtimeverification/casper-proofs/blob/master/Core/AccountableSafety.v>.

```

theory Casper
  imports Main
begin

```

Here we prove that the slashing conditions in Casper, as described at the url below by Vitalik Buterin, are such that if there is a fork, then one third of the validators are "slashed". <https://medium.com/@VitalikButerin/minimal-slashing-conditions-20f0b50>

We use first-order modeling as much as possible. This allows to reduce the size of the model, and also the size of the proofs from more than 1000 lines in Yoichi's proof to less than a 100.

```

locale byz-quorums =
  — "Here we fix two types 'q1 and 'q2 for quorums of cardinality greater than 2/3
  of the validators and quorum of cardinality greater than 1/3 of the validators.
  fixes member-1 :: 'n ⇒ 'q1 ⇒ bool (infix ∈1 50)
    — Membership in 2/3 set
    and member-2 :: 'n ⇒ 'q2 ⇒ bool (infix ∈2 50)
    — Membership in 1/3 set
  assumes ∧ q1 q2 . ∃ q3 . ∀ n . n ∈2 q3 ⟶ n ∈1 q1 ∧ n ∈1 q2
    — This is the only property of types 'q1 and 'q2 that we need: 2/3 quorums
    have 1/3 intersection

```

```

record ('n, 'h)state =
  — "'n is the type of validators (nodes), 'h hashes, and views are nat
  commit-msg :: 'n ⇒ 'h ⇒ nat ⇒ bool
  prepare-msg :: 'n ⇒ 'h ⇒ nat ⇒ nat ⇒ bool

```

```

locale casper = byz-quorums +
  — Here we make assumptions about hashes. In reality any message containing a
  hash not satisfying those should be dropped.
  fixes
    hash-parent :: 'h ⇒ 'h ⇒ bool (infix ← 50)
  assumes
    — "a hash has at most one parent which is not itself
    ∧ h1 h2 . h1 ← h2 ⟹ h1 ≠ h2
    and ∧ h1 h2 h3 . [h2 ← h1; h3 ← h1] ⟹ h2 = h3
begin

```

```

lemmas casper-assms-def = casper-def casper-axioms-def byz-quorums-def

```

```

inductive hash-ancestor (infix ←* 50) where
  h1 ← h2 ⟹ h1 ←* h2
  | [h1 ← h2; h2 ←* h3] ⟹ h1 ←* h3
declare hash-ancestor.intros[simp,intro]
lemma hash-ancestor-intro': assumes h1 ←* h2 and h2 ← h3 shows h1 ←* h3
  using assms by (induct h1 h2 rule:hash-ancestor.induct) auto

```

```

inductive nth-ancestor :: nat ⇒ 'h ⇒ 'h ⇒ bool where
  nth-ancestor 0 h1 h1

```

$\mid \llbracket nth\text{-ancestor } n \ h1 \ h2; h2 \leftarrow h3 \rrbracket \implies nth\text{-ancestor } (n+1) \ h1 \ h3$
declare $nth\text{-ancestor.intros[simp,intro]}$
inductive-cases $nth\text{-ancestor-succ}:nth\text{-ancestor } (n+1) \ h1 \ h3$
inductive-cases $zeroth\text{-ancestor}:nth\text{-ancestor } 0 \ h1 \ h3$
lemma $parent\text{-ancestor}:h1 \leftarrow h2 = nth\text{-ancestor } 1 \ h1 \ h2$
by ($metis \ One\text{-nat-def} \ add.\text{right-neutral} \ add\text{-Suc-right} \ add\text{-diff-cancel-left}' \ diff\text{-Suc-Suc} \ nat.simps(3) \ nth\text{-ancestor.simps}$)

All messages in epoch 0 are ignored; 0 is used as a special value (was -1 in the original model).

definition $prepared'$ **where**

$prepared' \ s \ q \ h \ v1 \ v2 \equiv v1 \neq 0 \wedge (\forall \ n . n \in_1 \ q \longrightarrow prepare\text{-msg } s \ n \ h \ v1 \ v2)$

definition $prepared$ **where**

$prepared \ s \ q \ h \ v1 \ v2 \equiv v1 \neq 0 \wedge v2 < v1 \wedge (\forall \ n . n \in_1 \ q \longrightarrow prepare\text{-msg } s \ n \ h \ v1 \ v2)$

definition $committed$ **where**

$committed \ s \ q \ h \ v \equiv v \neq 0 \wedge (\forall \ n . n \in_1 \ q \longrightarrow commit\text{-msg } s \ n \ h \ v)$

definition $fork$ **where**

$fork \ s \equiv \exists \ h1 \ h2 \ q1 \ q2 \ v1 \ v2 . committed \ s \ q1 \ h1 \ v1 \wedge committed \ s \ q2 \ h2 \ v2$
 $\wedge \neg(h2 \leftarrow^* h1 \vee h1 \leftarrow^* h2 \vee h1 = h2)$

definition $slashed-1$ **where** $slashed-1 \ s \ n \equiv$

$\exists \ h \ v . commit\text{-msg } s \ n \ h \ v \wedge (\forall \ q \ v2 . v2 < v \longrightarrow \neg prepared \ s \ q \ h \ v \ v2)$

definition $slashed-2$ **where**

$slashed-2 \ s \ n \equiv$

$\exists \ h \ v1 \ v2 . prepare\text{-msg } s \ n \ h \ v1 \ v2 \wedge v2 \neq 0 \wedge$

$(\forall \ v3 \ q \ h2 . v3 < v2 \longrightarrow \neg(nth\text{-ancestor } (v1 - v2) \ h2 \ h \wedge prepared \ s \ q \ h2 \ v2 \ v3))$

definition $slashed-3$ **where**

$slashed-3 \ s \ n \equiv$

$\exists \ h1 \ h2 \ v1 \ v2 \ v3 . v1 < v2 \wedge v3 < v1 \wedge$

$commit\text{-msg } s \ n \ h1 \ v1 \wedge prepare\text{-msg } s \ n \ h2 \ v2 \ v3$

definition $slashed-4$ **where**

$slashed-4 \ s \ n \equiv \exists \ h1 \ h2 \ v \ v1 \ v2 . (h1 \neq h2 \vee v1 \neq v2) \wedge$

$prepare\text{-msg } s \ n \ h1 \ v \ v1 \wedge prepare\text{-msg } s \ n \ h2 \ v \ v2$

definition $slashed$ **where** $slashed \ s \ n \equiv$

$slashed-1 \ s \ n \vee slashed-2 \ s \ n \vee slashed-3 \ s \ n \vee slashed-4 \ s \ n$

definition $one\text{-third-slashed}$ **where** $one\text{-third-slashed} \ s \equiv \exists \ q . \forall \ n . n \in_2 \ q \longrightarrow slashed \ s \ n$

lemmas $slashed\text{-defs} = slashed\text{-def} \ slashed-1\text{-def} \ slashed-2\text{-def} \ slashed-4\text{-def} \ slashed-3\text{-def} \ one\text{-third-slashed-def}$

lemmas $order\text{-defs} = class.linorder\text{-axioms-def} \ class.linorder\text{-def} \ class.order\text{-def} \ class.preorder\text{-def}$

$class.order\text{-axioms-def} \ class.order\text{-bot-def} \ class.order\text{-bot-axioms-def} \ linorder\text{-axioms}[where \ ?'a=nat]$

lemmas $casper\text{-defs} = slashed\text{-defs} \ prepared\text{-def} \ fork\text{-def} \ committed\text{-def} \ casper\text{-assms-def}$

lemma $l1$: **assumes** $prepared \ s \ q1 \ h1 \ v1 \ v2$ **and** $committed \ s \ q2 \ h2 \ v3$ **and** $v1 >$

```

v3 and  $\neg$ one-third-slashed s
  shows  $v1 > v2 \wedge v2 \geq v3$  using assms casper-axioms linorder-axioms[where
    ?'a=nat] unfolding casper-defs order-defs
  by metis

lemma l2: assumes nth-ancestor n h1 h2 and nth-ancestor m h2 h3
  shows nth-ancestor (n+m) h1 h3
  using assms
proof (induct m arbitrary: h1 h2 h3)
  case 0 then show ?case using nth-ancestor.cases by auto
next
  case (Suc m)
  then show ?case by (metis Suc-eq-plus1 add-Suc-right nth-ancestor.simps nth-ancestor-succ)
qed

lemma l3: assumes prepared s q1 h1 v1 v2 and committed s q2 h2 v3 and  $v1 > v3$ 
  and  $\neg$ one-third-slashed s
  shows nth-ancestor (v1 - v3) h2 h1
proof -
  show ?thesis using assms
proof (induct v1 - v3 arbitrary: v1 v2 v3 q1 q2 h1 h2 rule:less-induct)
  — "This is complete induction
  case less then show ?case
  proof (cases v1-v3=0)
    case True then show ?thesis using less.prem1 by linarith
  next
    case False then show ?thesis
  proof (cases v2=v3)
    case True
    obtain v4 q3 where 1:prepared s q3 h2 v2 v4
    by (metis True byz-quorums-axioms byz-quorums-def committed-def
      less.prem2 one-third-slashed-def slashed-1-def slashed-def)
    moreover have 3:h3 = h2  $\wedge$  v5 = v4 if prepared s q4 h3 v2 v5 for q4 h3 v5
    by (metis 1 byz-quorums-axioms byz-quorums-def less.prem4 one-third-slashed-def
      prepared-def slashed-4-def slashed-def that)
    ultimately show ?thesis using less.prem5 byz-quorums-axioms True
    by (simp only: casper-defs) metis
  next
    case False
    obtain q3 h3 v4 where 1:nth-ancestor (v1-v2) h3 h1 and 2:prepared s q3
      h3 v2 v4
    by (metis less.prem1-3 assms4 byz-quorums-axioms byz-quorums-def
      committed-def diff-is-0-eq diff-zero l1 one-third-slashed-def prepared-def slashed-2-def
      slashed-def)
    have 3:nth-ancestor (v2-v3) h2 h3
    by (metis False 2 l1 diff-less-mono less.hyps less.prem less-le)
    have 4:v1 > v2  $\wedge$  v2  $\geq$  v3 using assms casper.l1 casper-axioms less.prem1-3
  by metis
  show ?thesis using 1 3 4 less.prem3 l2 by force

```

qed
 qed
 qed
 qed

lemma *l4:assumes nth-ancestor n h1 h2 shows $h1 \leftarrow^* h2 \vee h1 = h2$ using assms*
proof (*induct n arbitrary:h1 h2*)
case 0 **then show** ?case **using** zeroth-ancestor **by** auto
next
case (Suc n)
obtain h3 **where** 1: $h3 \leftarrow h2$ **and** 2:*nth-ancestor n h1 h3* **using** nth-ancestor-succ
 Suc.prem **by** (metis Suc-eq-plus1)
show ?case **using** Suc.hyps[OF 2] 1 hash-ancestor-intro' **by** blast
qed

lemma *safety: assumes fork s shows one-third-slashed s*
proof –
obtain h1 h2 q1 q2 v1 v2 **where** 1:*committed s q1 h1 v1* **and** 2:*committed s q2 h2 v2*
and 3: $\neg (h2 \leftarrow^* h1 \vee h1 \leftarrow^* h2 \vee h1 = h2)$ **using** assms fork-def **by** blast
have 4: $\neg (\text{nth-ancestor } n \ h1 \ h2 \vee \text{nth-ancestor } n \ h2 \ h1 \vee h1 = h2)$ **for** n **using**
 l4 3 **by** auto
obtain q3 q4 v3 v4 **where** 5:*prepared s q3 h1 v1 v3* **and** 6:*prepared s q4 h2 v2*
 v4 **if** $\neg \text{one-third-slashed } s$ **using** 1 2
by (metis byz-quorums-axioms byz-quorums-def committed-def one-third-slashed-def
 slashed-1-def slashed-def)
show ?thesis
proof (cases $v1 = v2$)
case True
show ?thesis **using** 1 4 2 5 6 True casper-axioms unfolding byz-quorums-def
 casper-def slashed-def
 one-third-slashed-def casper-axioms committed-def prepared-def slashed-1-def
 slashed-4-def
by metis
next
case False **thus** ?thesis **using** l3 1 2 4 5 6 **by** (metis less-le not-less)
qed
qed
end
end