Verifying Finality for Blockchain Systems

Karl Palmskog* Milos Gligoric* Lucas Peña† Grigore Roşu†

*The University of Texas at Austin

†University of Illinois at Urbana-Champaign

Joint work with Brandon Moore at Runtime Verification, Inc.

Ethereum

- “a decentralized platform that runs smart contracts”
- accounts with balances instead of unspent transactions
- contracts execute in virtual machine on participating nodes

Blockchain Forks and Revisions

- "a blockchain diverges into two potential paths forward"
- accidental *or* intentional
- could be used by adversaries to control transactions

```
Fixpoint sprefixb (s1 s2 : seq block) :=
  if s2 is y :: s2' then
    if s1 is x :: s1' then (x == y) && (sprefixb s1' s2') else true
  else false.

Definition fork (bc1 bc2 : seq block) :=
  ~~[|| sprefixb bc1 bc2, sprefixb bc2 bc1 | bc1 == bc2].
```

Casper Finality

Buterin & Griffith, *Casper the Friendly Finality Gadget*, 2017

- overlay on top of an existing blockchain system
- "select[s] a unique chain which represents the canonical transactions of the ledger"
- protects against long-range revisions and crashes (assuming $> 2/3$ honest participants)

Background

ETHEREUM.ORG BUG BOUNTY PROGRAM ETHEREUM RESEARCH FORUM

Announcing Beneficiaries of the Ethereum Foundation Grants

Posted by Ethereum Team on March 7, 2018

Background

ETHEREUM.ORG BUG BOUNTY PROGRAM ETHEREUM RESEARCH FORUM

Announcing Beneficiaries of the Ethereum Foundation Grants

Posted by Ethereum Team on March 7, 2018

Awardee List

Here are the inaugural Ethereum Foundation grant winners:

L4 Research – Scalability Grant – $1.5M. State channels research.

Runtime Verification – Security Grant – $500K. Casper contract formal verification.

Background, Continued

Ethereum's Casper, Sharding Upgrades to Launch Together Allowing Better Scalability and Security

Omar Faridi
17 Jun 2018 / 337 views / In #Ethereum

- Ethereum co-founder Vitalik Buterin and the platform's development team may decide to launch Casper and Sharding upgrades together, instead of separately as planned earlier.
- New research has led Ethereum's developers to consider launching Ethereum's new proof-of-stake algorithm, Casper, through a shard (instead of a smart contract) to help reduce the cost of helping secure its network from 1,500 ETH to 32 ETH.

Casper Protocol Coq Formalization Goals

1. key claims in paper (following previous Isabelle/HOL models)
2. integration with blockchain model in Coq (Toychain)

Key Casper Notions

Validators and Votes

Validators deposit cryptocurrency (stake) and can then *vote* for blocks. With enough votes, a block becomes *finalized*. Validators who vote incorrectly get their deposits *slashed*.

Key Casper Notions

Validators and Votes

Validators deposit cryptocurrency (stake) and can then *vote* for blocks. With enough votes, a block becomes *finalized*. Validators who vote incorrectly get their deposits *slashed*.

Accountable Safety

Blocks in different block tree forks cannot both be finalized if more than 2/3 of validators *by deposit* behave honestly.

Key Casper Notions

Validators and Votes

Validators deposit cryptocurrency (stake) and can then *vote* for blocks. With enough votes, a block becomes *finalized*. Validators who vote incorrectly get their deposits *slashed*.

Accountable Safety

Blocks in different block tree forks cannot both be finalized if more than 2/3 of validators *by deposit* behave honestly.

Plausible Liveness

Regardless of what has happened before, it is always possible to continue to finalize blocks when more than 2/3 of validators *by deposit* follow the protocol.

Isabelle/HOL models and proofs by Hirai

On Older Casper Designs

- `DynamicValidatorSet.thy` is about two-message Casper (older) with a dynamic validator set (more realistic), and proves accountable safety (not plausible liveness).
- `Casper.thy` is about two-message Casper (older) with a static validator set (unrealistic), and proves accountable safety (not plausible liveness).
- `MinimumAlgo.thy` is about two-message Casper (older) with a dynamic validator set, and proves accountable safety and plausible liveness.

Isabelle/HOL models and proofs by Hirai

On Newer Casper Designs

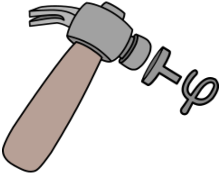
- `DynamicValidatorSetOneMessage.thy` is about one-message Casper (newer) with a dynamic validator set (more realistic), and proves accountable safety (not plausible liveness).
- `CasperOneMessage.thy` is about one-message Casper (newer) with a static validator set (unrealistic), and proves accountable safety (not plausible liveness).

Translating Between Proof Assistants

I was able to port many [HOL Light] proofs that I did not understand: despite the huge differences between the two proof languages, it was usually possible to guess what had to be proved from the HOL Light text, along with many key reasoning steps. Isabelle's automation was generally able to fill the gaps.

—L.C. Paulson, *Formalising Mathematics In Simple Type Theory*

Translating Between Proof Assistants, Continued



"[J]ust like the hammers for other systems, [CoqHammer] works very well for essentially first-order logic goals and becomes much less effective with other features of the logics [...]"

—L. Czajka & C. Kaliszyk, *Hammer for Coq*, 2018

Translating Between Proof Assistants, Continued

From `CasperOneMessage.thy`:

```
text {* 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 [...] *}
```

Isabelle/HOL to Coq via CoqHammer and MathComp

```
locale byz_quorums =
 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"
```

```
Variables quorum_1 quorum_2 : {set {set V}}.
Hypothesis qs : ∀ q1 q2, q1 ∈ quorum_1 → q2 ∈ quorum_1 →
 ∃ q3, q3 ∈ quorum_2 ∧ q3 ⊆ q1 ∧ q3 ⊆ q2.
```

Definitions

```
record ('n,'h)st = vote_msg :: "'n ⇒ 'h ⇒ nat ⇒ nat ⇒ bool"

locale casper = byz_quorums +
fixes
  hash_parent :: "'h ⇒ 'h ⇒ bool" (infix "←" 50)
fixes
  genesis :: 'h
assumes
  "⋀ h1 h2 . h1 ← h2 ⟹ h1 ≠ h2"
  and "⋀ h1 h2 h3 . ⟨h2 ← h1; h3 ← h1⟩ ⟹ h2 = h3"
```

```
Record st := { vote_msg : Validator → Hash → nat → nat → bool }.

Variable hash_parent : rel Hash.
Notation "h1 ← h2" := (hash_parent h1 h2) (at level 50).
Variable genesis : Hash.
Hypothesis hash_at_most_one_parent : ∀ h1 h2 h3,
 (h2 ← h1) → (h3 ← h1) → h2 = h3.
```

Definitions, Continued

```
definition justified_link
where
"justified_link s q parent pre new now ≡
(∀ n. n ∈₁ q ⟶ vote_msg s n new now pre) ∧
 nth_ancestor (now - pre) parent new ∧
 now > pre"
```

```
Definition justified_link s q parent pre new now :=
  q ∈ quorum_1 ∧ (∀ n, n ∈ q → vote_msg s n new now pre) ∧
  nth_ancestor (now - pre) parent new ∧
  now > pre.
```

Lemmas and Induction Proofs

```
lemma non_equal_case_ind:
 assumes "justified s h1 v1"
 assumes "finalized s q2 h2 v2 xa"
 assumes "¬ h2 ←* h1"
 assumes "h1 ≠ h2"
 assumes "v1 > v2"
 shows "one_third_slashed s"
using assms proof
 (induct "v1 - v2" arbitrary: h1 v1 rule:less_induct)
```

```
Lemma non_equal_case_ind : ∀ s h1 v1 q2 h2 v2 xa,
  justified s h1 v1 →
  finalized s q2 h2 v2 xa →
  h2 </~* h1 →
  h1 ≠ h2 →
  v1 > v2 →
  one_third_slashed s.
```

Lemmas and Induction Proofs, Continued

```
From mathcomp Require Import all_ssreflect.

Section StrongInductionLtn.

Variable P : nat → Prop.
Hypothesis IH : ∀ m, (∀ n, n < m → P n) → P m.

Lemma P0 : P 0.

Lemma pred_increasing : ∀ (n m : nat), n <= m → n.-1 <= m.-1.

Local Lemma strong_induction_all : ∀ n, (∀ m, m <= n → P m).

Theorem strong_induction_ltn : ∀ n, P n.

End StrongInductionLtn.
```

Accountable Safety

```
Definition finalization_fork s :=
  ∃ h1 h2 q1 q2 v1 v2 c1 c2,
    finalized s q1 h1 v1 c1 ∧
    finalized s q2 h2 v2 c2 ∧
    h2 </~* h1 ∧ h1 </~* h2 ∧ h1 ≠ h2.

(* validators mustn't double vote or vote in same span *)
Definition slashed s n : Prop :=
  slashed_dbl_vote s n ∨ slashed_surround s n.

Definition quorum_slashed s :=
  ∃ q, q ∈ quorum_2 ∧ ∀ n, n ∈ q → slashed s n.

Theorem accountable_safety : ∀ s,
  finalization_fork s → quorum_slashed s.
```

Plausible Liveness

- Isabelle/HOL proofs only for old Casper (two message types)
- recent Casper removed all slashing conditions which depended on the state of the chain when vote was made
- one of these conditions was essential to the proof
- details in our tech report!

Connecting Model to Paper Claims

```
Variables (T : finType) (d : T → nat) (x y z : nat).

Definition gdset n : {set {set T}} :=
 [set s in powerset [set: T] | \sum_(t in s) (d t) >= n].

Lemma gt_dset_in : ∀ n (s : {set T}),
 \sum_(t in s) (d t) >= n = (s ∈ gdset n).

Local Notation bot := (((x * \sum_(t : T) (d t)) %/ y).+1).
Local Notation top := (((z * \sum_(t : T) (d t)) %/ y).+1).

Hypothesis constr : bot + \sum_(t : T) (d t) <= 2 * top.

Lemma d_bot_top_intersection :
  ∀ q1 q2, q1 ∈ gdset top → q2 ∈ gdset top →
  ∃ q3, q3 ∈ gdset bot ∧ q3 ⊆ q1 ∧ q3 ⊆ q2.
```

Connecting Models to Paper Claims, Continued

```
Lemma constr_thirds : ∀ n, (n %/ 3).+1 + n <= 2 * (2 * n %/ 3).+1.

Variables (Validator : finType) (deposit : Validator → nat).

Definition deposits := \sum_(v : Validator) (deposit v).
Definition deposit_bot := gdset deposit (deposits %/ 3).+1.
Definition deposit_top := gdset deposit ((2 * deposits) %/ 3).+1.

Lemma Validators_deposit_constr_thirds :
 ((1 * deposits) %/ 3).+1 + deposits <= 2 * ((2 * deposits) %/ 3).+1.
Proof. by rewrite mul1n; apply: constr_thirds. Qed.

Lemma deposit_bot_top_validator_intersection :
  ∀ q1 q2, q1 ∈ deposit_top → q2 ∈ deposit_top →
  ∃ q3, q3 ∈ deposit_bot ∧ q3 ⊆ q1 ∧ q3 ⊆ q2.
```

Instantiating Block Hashes via Toychain

```
Definition Blocktree := union_map Hash Block.

Definition hash_parent (bt : Blocktree) : rel Hash :=
 [rel x y | (x ∈ dom bt) && if find y bt is Some b
 then parent_hash b == x else false].
```

Current and Future Work

- dynamic validator sets
- validator deposits and slashes
- capturing beacon chain and shards chains explicitly

Translation Experience

> *All existing proof translation techniques work by emulating one calculus within another at the level of primitive inferences. Could proofs instead be translated at the level of a mathematical argument?*

—L.C. Paulson, *Formalising Mathematics In Simple Type Theory*

Coq/Ssreflect and MathComp Experience

- definitions more important than proof language
- library of blockchain data structures would be useful
- missed `omega` tactic, but see MathComp issue #251
- using bigops was hard at first, but paid off

Conclusion

Casper verification is WIP; future depends on Ethereum foundation goals and decisions

- Contact me: `palmskog@utexas.edu`, `https://setoid.com`
- Coq proofs and tech report: `https://github.com/runtimeverification/casper-proofs`
- Isabelle/HOL proofs: `https://github.com/palmskog/pos`
- Tech report has more details, e.g., on plausible liveness