A small PoS protocol with dynamic validator set

Yoichi Hirai*

March 19, 2017

Contents

*i@yoichihirai.com

- This is a formalization attempt of Vitalik Buterin "Safety Under Dynamic Validator Sets" https://medium.com/@VitalikButerin/safety-under-dynamic-validator-s .8eojzbyou
- However, it is not a faithful formalization currently. I defined forks using the `prepare_src` aware chains, I didn't need slashing condidions one and two.
- This document is produced from the code available at https://github.com/pirapira/pos.
- To get updates on this project and similar ones, follow http://gitter.im/ethereum/formal-methods or https://gitter.im/ethereum/research.

theory *DynamicValidatorSet*

imports *Main*

begin

1 Definitions Necessary to Understand Accountable Safety (not skippable)

In this development we do not know much about hashes. There are many hashes. Two hashes might be equal or not.

datatype *hash* = *Hash int*

Views are numbers. We actually need the fact that views are lines up in a total order. Otherwise accountable safety can be broken.

type-synonym *view* = *int*

We have two kinds of messages. A Commit message contains a hash and a view. A prepare message contains a hash and two views. At this point a message is not signed by anybody.

datatype *message* =
  *Commit hash * view*
| *Prepare hash * view * view*

We need a set of validators. Here, we just define a datatype containing infinitely many validators. Afterwards, when we look at a particular situation, the situation would contain a finite set of validators.

datatype *validator* = *Validator int*

A message signed by a validator can be represented as a pair of a validator and a message.

type-synonym *signed-message* = *validator* $*$ *message*

Almost everything in this document depends on situations. A situation contains a set of validators, a set of signed messages, and a function specifying parents of hashes.

A situation might be seen from a global point of view where every sent messages can be seen, or more likely seen from a local point of view.

record *situation* =
RearValidators :: *hash* ⇒ *validator set*
FwdValidators :: *hash* ⇒ *validator set*
Messages :: *signed-message set*
PrevHash :: *hash* ⇒ *hash option*

In the next section, we are going to determine which of the validators are slashed in a situation.

A hash's previous hash's previous hash is a second-ancestor. Later, we will see that an honest validator will send a prepare message only after seeing enough prepare messages for an ancestor of a particular degree.

fun *nth-ancestor* :: *situation* ⇒ *nat* ⇒ *hash* ⇒ *hash option*
where
nth-ancestor - 0 h = *Some h*
| *nth-ancestor s* (*Suc n*) *h* =
(*case PrevHash s h of*
None ⇒ *None*
| *Some h'* ⇒ *nth-ancestor s n h'*)

In the slashing condition, we will be talking about two-thirds of the validators doing something.

We can lift any predicate about a validator into a predicate about a situation: two thirds of the validators satisfy the predicate.

definition *two-thirds* :: *validator set* ⇒ (*validator* ⇒ *bool*) ⇒ *bool*
where
two-thirds vs f =
$(2 * card\ vs \leq 3 * card\ (\{n.\ n \in vs \wedge f\ n\}))$

Similarly for one-third, more-than-two-thirds, and more-than-one-third.

definition *one-third* :: *validator set* ⇒ (*validator* ⇒ *bool*) ⇒ *bool*
where
one-third vs f =
$(card\ vs \leq 3 * card\ (\{n.\ n \in vs \wedge f\ n\}))$

It matters when two-thirds of validators are saying something.

definition *two-thirds-sent-message* :: *situation* ⇒ *validator set* ⇒ *message* ⇒ *bool*
where
two-thirds-sent-message s vs m =
  two-thirds vs (λ *n*. (*n*, *m*) ∈ *Messages s*)

A hash is prepared when two-thirds of the validators have sent a certain message.

definition *prepared* :: *situation* ⇒ *validator set* ⇒ *hash* ⇒ *view* ⇒ *view* ⇒ *bool*
where
prepared s vs h v vsrc =
  (*two-thirds-sent-message s vs* (*Prepare* (*h*, *v*, *vsrc*)))

A hash is committed when two-thirds of the validators have sent a certain message.

definition *committed* :: *situation* ⇒ *validator set* ⇒ *hash* ⇒ *view* ⇒ *bool*
where
committed s vs h v = *two-thirds-sent-message s vs* (*Commit* (*h*, *v*))

1.1 Prepare Messages' Sources

As we will see, honest validators should send a prepare message only when it has enough prepare messages at a particular view. Those prepare messages need to be signed by two-thirds of both the rear and the forward validators.

A hash at a view and a view source is prepared by the rear validators when two-thirds of the rear validators have signed the prepare message.

definition *prepared-by-rear* :: *situation* ⇒ *hash* ⇒ *view* ⇒ *view* ⇒ *bool*
where
prepared-by-rear s h v vsrc =
  (*prepared s* (*RearValidators s h*) *h v vsrc*)

Similarly for the forward validators.

definition *prepared-by-fwd* :: *situation* ⇒ *hash* ⇒ *view* ⇒ *view* ⇒ *bool*
where
prepared-by-fwd s h v vsrc =
  (*prepared s* (*FwdValidators s h*) *h v vsrc*)

When both of these happens, a prepare is signed by both the rear and the forward validator sets.

definition *prepared-by-both* :: *situation* ⇒ *hash* ⇒ *view* ⇒ *view* ⇒ *bool*
where
prepared-by-both s h v vsrc =
  (*prepared-by-rear s h v vsrc* ∧ *prepared-by-fwd s h v vsrc*)

Similar definitions for commit messages.

definition *committed-by-rear* :: *situation* ⇒ *hash* ⇒ *view* ⇒ *bool*
where

committed-by-rear s h v =
  (*committed s* (*RearValidators s h*) *h v*)

definition *committed-by-fwd* :: *situation* ⇒ *hash* ⇒ *view* ⇒ *bool*
where
committed-by-fwd s h v =
  (*committed s* (*FwdValidators s h*) *h v*)

definition *committed-by-both* :: *situation* ⇒ *hash* ⇒ *view* ⇒ *bool*
where
committed-by-both s h v =
  (*committed-by-rear s h v* ∧ *committed-by-fwd s h v*)

One type of prepare source is the normal one. The normal source needs to have the same rear validator set and the same forward validator set.

definition *validators-match* :: *situation* ⇒ *hash* ⇒ *hash* ⇒ *bool*
where
validators-match s h0 h1 =
 (*RearValidators s h0* = *RearValidators s h1* ∧
  FwdValidators s h0 = *FwdValidators s h1*)

fun *sourcing-normal* :: *situation* ⇒ *hash* ⇒ (*hash* × *view* × *view*) ⇒ *bool*
where
sourcing-normal s h (h', v', *v-src*) =
 (∃ *v-ss*.
  prepared-by-both s h v-src v-ss ∧
  −*1* ≤ *v-src* ∧
  v-src < v' ∧
  nth-ancestor s (*nat* (v' − *v-src*)) h' = *Some h* ∧
  validators-match s h h')

Another type of sourcing allows changing the validator sets. The forward validator set of the source needs to coincide with the rear validator set of the newly prepared hash.

definition *validators-change* :: *situation* ⇒ *hash* ⇒ *hash* ⇒ *bool*
where
validators-change s ancient next =
  (*FwdValidators s ancient* = *RearValidators s next*)

fun *sourcing-switching-validators* ::
situation ⇒ *hash* ⇒ (*hash* × *view* × *view*) ⇒ *bool*
where
sourcing-switching-validators s h (h', v', *v-src*) =
 (∃ *v-ss*.
  prepared-by-both s h v-src v-ss ∧
  committed-by-both s h v-src ∧
  −*1* ≤ *v-src* ∧
  v-src < v' ∧
  nth-ancestor s (*nat* (v' − *v-src*)) h' = *Some h* ∧

validators-change s h h′)

A prepare message's source needs to be one of these two types.

definition *sourcing* :: *situation* ⇒ *hash* ⇒ (*hash* × *view* × *view*) ⇒ *bool*
where
sourcing s h-new tri = (*sourcing-normal s h-new tri* ∨ *sourcing-switching-validators s h-new tri*)

1.2 Slashing Conditions

In a situation, a validator might be slashed or not. A validator is slashed individually although later we will be often talking "unless one-third of the validators are slashed."

[iii] A validator is slashed when it has sent a commit message and a prepare message containing view numbers in a specific constellation.

definition *slashed-three* :: *situation* ⇒ *validator* ⇒ *bool*
where
slashed-three s n =
  (∃ *x y v w u*.
   (*n*, *Commit* (*x*, *v*)) ∈ *Messages s* ∧
   (*n*, *Prepare* (*y*, *w*, *u*)) ∈ *Messages s* ∧
   u < *v* ∧ *v* < *w*)

[iv] A validator is slashed when it has signed two different Prepare messages at the same view.

definition *slashed-four* :: *situation* ⇒ *validator* ⇒ *bool*
where
slashed-four s n =
  (∃ *x1 x2 v vs1 vs2*.
   (*n*, *Prepare* (*x1*, *v*, *vs1*)) ∈ *Messages s* ∧
   (*n*, *Prepare* (*x2*, *v*, *vs2*)) ∈ *Messages s* ∧
   (*x1* ≠ *x2* ∨ *vs1* ≠ *vs2*))

A validator is slashed when at least one of the above conditions [i]–[iv] hold.

definition *slashed* :: *situation* ⇒ *validator* ⇒ *bool*
where
slashed s n = (*slashed-three s n* ∨
         slashed-four s n)

definition *one-third-slashed* :: *situation* ⇒ *validator set* ⇒ *bool*
where
one-third-slashed s vs = *one-third vs* (*slashed s*)

However, since it does not make sense to divide the cardinality of an infinite set by three, we should be talking about situations where the set of validators is finite.

definition *one-third-of-rear-slashed* :: *situation* ⇒ *hash* ⇒ *bool*
where
one-third-of-rear-slashed s h = *one-third* (*RearValidators s h*) (*slashed s*)

definition *one-third-of-fwd-slashed* :: *situation* ⇒ *hash* ⇒ *bool*
where
one-third-of-fwd-slashed s h = *one-third* (*FwdValidators s h*) (*slashed s*)

1.3 Validator History Tracking

In the statement of accountable safety, we need to be a bit specific about which validator set the slashed validators belong to. A singleton is also a validator set and the 2/3 of a random singleton being slashed should not be significant. So, when we have a fork, we start from the root of the fork and identify the heirs of the initial validator sets. Our statement says 2/3 of a heir validator set are slashed.

There are two ways of inheriting the title of relevant validator set. These correspond to the two ways of sourcing a prepare message.

fun *inherit-normal* :: *situation* ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ *bool*
where
inherit-normal s (*h-old*, *v-src*) (*h-new*, *v*) =
  (*prepared-by-both s h-new v v-src* ∧ *sourcing-normal s h-old* (*h-new*, *v*, *v-src*))

lemma *inherit-normal-view-increase* :
  inherit-normal s (*h-old*, *v-src*) (*h-new*, *v*) ⟹
  (*v-src* < *v*)
⟨*proof*⟩

fun *inherit-switching-validators* ::
  situation ⇒ (*hash* × *view*) ⇒
    (*hash* × *view*) ⇒ *bool*
where
inherit-switching-validators s (*h-old*, *v-old*) (*h-new*, *v-new*) =
  (*prepared-by-both s h-new v-new v-old* ∧
  sourcing-switching-validators s h-old (*h-new*, *v-new*, *v-old*))

The heir relation is just zero-or-more repetition of the inheritance.

inductive *heir* :: *situation* ⇒
    (*hash* × *view*) ⇒
    (*hash* × *view*) ⇒ *bool*
where
  heir-self : *prepared-by-both s h v v-src* ⟹ *heir s* (*h*, *v*) (*h*, *v*)
| *heir-normal-step* : *heir s* (*h*, *v*) (*h*′, *v*′) ⟹
    inherit-normal s (*h*′, *v*′) (*h*″, *v*″) ⟹
    heir s (*h*, *v*) (*h*″, *v*″)
| *heir-switching-step* : *heir s* (*h*, *v*) (*h*′, *v*′) ⟹
    inherit-switching-validators s (*h*′, *v*′) (*h*″, *v*″) ⟹

$heir\ s\ (h,\ v)\ (h'',\ v'')$

When two hashes are not in the inheritance relation in either direction, the two hashes are not on the same heir chain. In the statement of accountable safety, we use this concept to detect conflicts (which should not happen until 2/3 of a legitimate heir are slashed).

definition *on-same-heir-chain* :: *situation* $\Rightarrow$ (*hash* $\times$ *view*) $\Rightarrow$ (*hash* $\times$ *view*) $\Rightarrow$ *bool*
where
on-same-heir-chain s x y = (*heir s x y* $\vee$ *heir s y x*)

When heirs are not on the same chain of legitimacy, they have forked.

fun *fork* :: *situation* $\Rightarrow$
(*hash* $\times$ *view*) $\Rightarrow$
(*hash* $\times$ *view*) $\Rightarrow$
(*hash* $\times$ *view*) $\Rightarrow$ *bool*
where
fork s (*root*, *v*) (*h1*, *v1*) (*h2*, *v2*) =
($\neg$ *on-same-heir-chain s* (*h1*, *v1*) (*h2*, *v2*) $\wedge$ *heir s* (*root*, *v*) (*h1*, *v1*) $\wedge$ *heir s* (*root*, *v*) (*h2*, *v2*))

A fork is particularly bad when the end points are committed, not only prepared.

fun *fork-with-commits* :: *situation* $\Rightarrow$ (*hash* $\times$ *view*) $\Rightarrow$ (*hash* $\times$ *view*) $\Rightarrow$ (*hash* $\times$ *view*) $\Rightarrow$ *bool*
where
fork-with-commits s (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) =
(*fork s* (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) $\wedge$
committed-by-both s h v $\wedge$
committed-by-both s h1 v1 $\wedge$
committed-by-both s h2 v2)

2 Auxiliary Things (skippable)

2.1 Sets and Arithmetics

lemma *sum-suc-disj* :
n-one + *n-two* $\leq$ *Suc k* $\Longrightarrow$
n-one + *n-two* $\leq$ *k* $\vee$
n-one + *n-two* = *Suc k*
$\langle$*proof*$\rangle$

lemma *sum-eq-disj* :
((*n-one* :: *nat*) $\leq$ *1* $\wedge$ (*n-two* :: *nat*) $\leq$ *1*) $\vee$
n-one > *1* $\vee$ *n-two* > *1*

$\langle$*proof*$\rangle$

lemma *sum-eq-case1* :
$n\text{-}one + n\text{-}two = Suc\ k \Longrightarrow$
$n\text{-}one > 1 \Longrightarrow$
$\exists\ n\text{-}one\text{-}pre.\ n\text{-}one\text{-}pre \geq 1 \wedge n\text{-}one = Suc\ n\text{-}one\text{-}pre \wedge n\text{-}one\text{-}pre + n\text{-}two = k$
⟨*proof*⟩

lemma *sum-eq-case2* :
$n\text{-}one + n\text{-}two = Suc\ k \Longrightarrow$
$n\text{-}two > 1 \Longrightarrow$
$\exists\ n\text{-}two\text{-}pre.\ n\text{-}two\text{-}pre \geq 1 \wedge n\text{-}two = Suc\ n\text{-}two\text{-}pre \wedge n\text{-}one + n\text{-}two\text{-}pre = k$
⟨*proof*⟩

lemma *sum-suc* :
$n\text{-}one + n\text{-}two \leq Suc\ k \Longrightarrow$
$n\text{-}one + n\text{-}two \leq k \vee$
$n\text{-}one \leq 1 \wedge n\text{-}two \leq 1 \vee$
$(\exists\ n\text{-}one\text{-}pre.\ n\text{-}one\text{-}pre \geq 1 \wedge n\text{-}one = Suc\ n\text{-}one\text{-}pre \wedge n\text{-}one\text{-}pre + n\text{-}two =$
$k) \vee$
$(\exists\ n\text{-}two\text{-}pre.\ n\text{-}two\text{-}pre \geq 1 \wedge n\text{-}two = Suc\ n\text{-}two\text{-}pre \wedge n\text{-}one + n\text{-}two\text{-}pre =$
$k)$

⟨*proof*⟩

lemma *card-not* [*simp*] :
$finite\ s \Longrightarrow$
$card\ \{n \in s.\ \neg\ f\ n\} = card\ s - card\ \{n \in s.\ f\ n\}$
⟨*proof*⟩

lemma *set-conj* [*simp*] :
$\{n \in s.\ f\ n \wedge g\ n\} = \{n \in s.\ f\ n\} \cap \{n \in s.\ g\ n\}$
⟨*proof*⟩

lemma *two-more-two-set* :
$finite\ s \Longrightarrow$
$2 * card\ s \leq 3 * card\ \{n \in s.\ f\ n\} \Longrightarrow$
$2 * card\ s < 3 * card\ \{n \in s.\ g\ n\} \Longrightarrow$
$card\ s$
$< 3 * card\ (\{n \in s.\ f\ n\} \cap \{n \in s.\ g\ n\})$

⟨*proof*⟩

lemma *card-nonzero-exists* :
$card\ \{n \in s.\ f\ n\} > 0 \Longrightarrow$
$\exists\ n \in s.\ f\ n$

$\langle proof \rangle$

lemma *card-conj-le* :
$finite\ s \Longrightarrow$
$card\ (\{n \in s.\ f\ n\} \cap \{n \in s.\ g\ n\})$
$= card\ \{n \in s.\ f\ n\} + card\ \{\ n \in s.\ g\ n\} - card\ (\{n \in s.\ f\ n\} \cup \{n \in s.\ g\ n\})$
$\langle proof \rangle$

lemma *two-two-set* :
$2 * card\ s \leq 3 * card\ \{n \in s.\ f\ n\} \Longrightarrow$
$2 * card\ s \leq 3 * card\ \{n \in s.\ g\ n\} \Longrightarrow$
$finite\ s \Longrightarrow$
$card\ s \leq 3 * card\ (\{n \in s.\ f\ n\} \cap \{n \in s.\ g\ n\})$
$\langle proof \rangle$

lemma *inclusion-card-le* :
$\forall n.\ n \in Validators\ s \longrightarrow f\ n \longrightarrow g\ n \Longrightarrow$
$finite\ (Validators\ s) \Longrightarrow$
$card\ \{n \in Validators\ s.\ f\ n\} \leq card\ \{n \in Validators\ s.\ g\ n\}$
$\langle proof \rangle$

lemma *nat-min-min* :
$vs1 < v \Longrightarrow$
$\neg\ vs1 < c\text{-}view \Longrightarrow$
$(nat\ (v - vs1) + nat\ (vs1 - c\text{-}view)) = nat\ (v - c\text{-}view)$
$\langle proof \rangle$

lemma *view-total* [*simp*]:
$(v2 :: view) \leq v1 \lor v1 \leq v2$
$\langle proof \rangle$

lemma *sum-is-suc-dest* :
$Suc\ n = n1 + n2 \Longrightarrow$
$((\exists\ n1'.\ n1 = Suc\ n1' \land n = n1' + n2) \lor$
$(\exists\ n2'.\ n2 = Suc\ n2' \land n = n1 + n2'))$

$\langle proof \rangle$

lemma *find-max-ind-step* :
$\forall u.\ n = nat\ (u - s) \longrightarrow s \in (S :: int\ set) \longrightarrow (\forall x.\ x \in S \longrightarrow x \leq u)$
$\longrightarrow (\exists m.\ m \in S \land (\forall y{>}m.\ y \notin S)) \Longrightarrow$
$Suc\ n = nat\ (u - s) \Longrightarrow s \in S \Longrightarrow \forall x.\ x \in S \longrightarrow x \leq u \Longrightarrow \exists m.\ m \in S \land$
$(\forall y{>}m.\ y \notin S)$
$\langle proof \rangle$

lemma *find-max-ind* :
$\forall\ u.$
$n = nat\ (u - s) \longrightarrow$
$s \in (S :: int\ set) \longrightarrow$
$(\forall\ x.\ x \in S \longrightarrow x \leq u) \longrightarrow$
$(\exists\ m.\ m \in S \wedge$
$(\forall\ y.\ y > m \longrightarrow y \notin S))$
$\langle proof \rangle$

lemma *find-max* :
$s \in (S :: int\ set) \Longrightarrow$
$\forall\ x.\ x \in S \longrightarrow x \leq u \Longrightarrow$
$\exists\ m.\ m \in S \wedge$
$(\forall\ y.\ y > m \longrightarrow y \notin S)$
$\langle proof \rangle$

lemma *one-third-mp* :
finite $X \Longrightarrow$
$\forall\ v.\ p\ v \longrightarrow q\ v \Longrightarrow$
one-third $X\ p \Longrightarrow$ *one-third* $X\ q$
$\langle proof \rangle$

lemma *two-thirds-two-thirds-one-third* :
finite $X \Longrightarrow$
two-thirds $X\ p \Longrightarrow$
two-thirds $X\ q \Longrightarrow$
one-third $X\ (\lambda\ x.\ p\ x \wedge q\ x)$

$\langle proof \rangle$

2.2 Validator History Tracking

lemma *heir-increases-view* :
heir $s\ t\ t' \Longrightarrow$ *snd* $t \leq$ *snd* t'
$\langle proof \rangle$

inductive *heir-after-n-switching* ::
$nat \Rightarrow situation \Rightarrow$
$(hash \times view) \Rightarrow$
$(hash \times view) \Rightarrow bool$
where
heir-n-self : *prepared-by-both* $s\ h\ v$ *v-src* $\Longrightarrow$ *heir-after-n-switching* $0\ s\ (h,\ v)\ (h,\ v)$
| *heir-n-normal-step* :
heir-after-n-switching $n\ s\ (h,\ v)\ (h',\ v') \Longrightarrow$
inherit-normal $s\ (h',\ v')\ (h'',\ v'') \Longrightarrow$
heir-after-n-switching $n\ s\ (h,\ v)\ (h'',\ v'')$

| *heir-n-switching-step* :
heir-after-n-switching $n\ s\ (h,\ v)\ (h',\ v') \Longrightarrow$
inherit-switching-validators $s\ (h',\ v')\ (h'',\ v'') \Longrightarrow$
heir-after-n-switching $(Suc\ n)\ s\ (h,\ v)\ (h'',\ v'')$

lemma *inherit-switching-validators-increase-view* :
inherit-switching-validators s (*h-old*,*v-old*) (*h-new*, *v-new*) $\Longrightarrow$
v-old $<$ *v-new*
⟨*proof*⟩

lemma *every-heir-is-after-n-switching* :
heir $s\ p0\ p1 \Longrightarrow \exists\ n.$ *heir-after-n-switching* $n\ s\ p0\ p1$
⟨*proof*⟩

fun *fork-with-n-switching* :: *situation* $\Rightarrow$
$(hash \times view) \Rightarrow$
$nat \Rightarrow (hash \times view) \Rightarrow$
$nat \Rightarrow (hash \times view) \Rightarrow bool$
where
fork-with-n-switching
$s\ (root,\ v)\ n1\ (h1,\ v1)\ n2\ (h2,\ v2) =$
$(\neg$ *on-same-heir-chain* $s\ (h1,\ v1)\ (h2,\ v2)\ \wedge$
heir-after-n-switching $n1\ s\ (root,\ v)\ (h1,\ v1)\ \wedge$
heir-after-n-switching $n2\ s\ (root,\ v)\ (h2,\ v2))$

lemma *fork-has-n-switching* :
fork $s\ (r,\ v)\ (h1,\ v1)\ (h2,\ v2) \Longrightarrow$
$\exists\ n1\ n2.$ *fork-with-n-switching* $s\ (r,\ v)\ n1\ (h1,\ v1)\ n2\ (h2,\ v2)$
⟨*proof*⟩

lemma *heir-decomposition* :
heir $s\ (h,\ v)\ (h'',\ v'') \Longrightarrow$
$((\exists$ *v-src*. $h = h'' \wedge v = v'' \wedge$ *prepared-by-both* $s\ h\ v$ *v-src*$)\ \vee$
$(\exists\ h'\ v'.$ *heir* $s\ (h,\ v)\ (h',\ v') \wedge$ *inherit-normal* $s\ (h',\ v')\ (h'',\ v''))\ \vee$
$(\exists\ h'\ v'.$ *heir* $s\ (h,\ v)\ (h',\ v') \wedge$ *inherit-switching-validators* $s\ (h',\ v')\ (h'',\ v''))$
)
⟨*proof*⟩

lemma *heir-same-height* :
heir $s\ (h',\ v)\ (h,\ v) \Longrightarrow$
$h' = h$
⟨*proof*⟩

fun *fork-with-center* :: *situation* ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ *bool*
where
fork-with-center s (*h-orig*, *v-orig*) (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) =
(*fork s* (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) ∧
heir s (*h-orig*, *v-orig*) (*h*, *v*) ∧ (∗ *This is used to connect the whole setup with the original statement* ∗)
committed-by-both s h v ∧
committed-by-both s h1 v1 ∧
committed-by-both s h2 v2)

fun *fork-with-center-with-n-switching* :: *situation* ⇒ (*hash* × *view*) ⇒
(*hash* × *view*) ⇒ *nat* ⇒ (*hash* × *view*) ⇒ *nat* ⇒ (*hash* × *view*) ⇒ *bool*
where
fork-with-center-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) *n1* (*h1*, *v1*) *n2* (*h2*, *v2*) =
(*fork-with-n-switching s* (*h*, *v*) *n1* (*h1*, *v1*) *n2* (*h2*, *v2*) ∧
heir s (*h-orig*, *v-orig*) (*h*, *v*) ∧ (∗ *This is used to connect the whole setup with the original statement* ∗)
committed-by-both s h v ∧
committed-by-both s h1 v1 ∧
committed-by-both s h2 v2)

lemma *fork-with-center-has-n-switching* :
fork-with-center s (*h-orig*, *v-orig*) (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) ⟹
∃ *n1 n2*.
fork-with-center-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) *n1* (*h1*, *v1*) *n2* (*h2*, *v2*)
⟨*proof*⟩

fun *fork-root-views* :: *situation* ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ *view set*
where
fork-root-views s (*h-orig*, *v-orig*) (*h1*, *v1*) (*h2*, *v2*) =
{ *v*. (∃ *h*. *fork-with-center s* (*h-orig*, *v-orig*) (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*)) }

fun *fork-with-center-with-high-root* ::
situation ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ *bool*
where
fork-with-center-with-high-root s (*h-orig*, *v-orig*) (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) =
(*fork-with-center s* (*h-orig*, *v-orig*) (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) ∧
(∀ h' v'. $v' > v$ ⟶
¬ *fork-with-center s* (*h-orig*, *v-orig*) (h', v') (*h1*, *v1*) (*h2*, *v2*)))

fun *fork-with-center-with-high-root-with-n-switching* ::
situation ⇒ (*hash* × *view*) ⇒ (*hash* × *view*) ⇒ *nat* ⇒ (*hash* × *view*) ⇒
nat ⇒ (*hash* × *view*) ⇒ *bool*

where
fork-with-center-with-high-root-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) *n1* (*h1*, *v1*) *n2* (*h2*, *v2*) =
(*fork-with-center-with-n-switching s* (*h-orig*, *v-orig*) (*h*, *v*) *n1* (*h1*, *v1*) *n2* (*h2*, *v2*) ∧
(∀ *h′ v′*. *v′* > *v* ⟶
¬ *fork-with-center s* (*h-orig*, *v-orig*) (*h′*, *v′*) (*h1*, *v1*) (*h2*, *v2*)))

lemma *fork-with-center-with-high-root-has-n-switching* :
fork-with-center-with-high-root s (*h-orig*, *v-orig*) (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) ⟹
∃ *n1 n2*.
fork-with-center-with-high-root-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) *n1* (*h1*, *v1*) *n2* (*h2*, *v2*)
⟨*proof*⟩

lemma *fork-with-center-choose-high-root* :
fork-with-center s (*h-orig*, *v-orig*) (*h*, *v*) (*h1*, *v1*) (*h2*, *v2*) ⟹
∃ *h′ v′*. *fork-with-center-with-high-root s* (*h-orig*, *v-orig*) (*h′*, *v′*) (*h1*, *v1*) (*h2*, *v2*)
⟨*proof*⟩

lemma *forget-number-of-switching*:
heir-after-n-switching n s (*h-twoa*, *v-twoa*) (*h-one*, *v-one*)
⟹ *heir s* (*h-twoa*, *v-twoa*) (*h-one*, *v-one*)
⟨*proof*⟩

lemma *inherit-normal-means-heir* :
inherit-normal s (*h′*, *v′*) (*h″*, *v″*) ⟹
heir s (*h′*, *v′*) (*h″*, *v″*)
⟨*proof*⟩

lemma *chain-and-inherit* :
inherit-normal s (*h′*, *v′*) (*h″*, *v″*) ⟹
v-two ≤ *snd* (*h″*, *v″*) ⟹
¬ *on-same-heir-chain s* (*h″*, *v″*) (*h-two*, *v-two*) ⟹
v-two ≤ *snd* (*h′*, *v′*) ⟹
on-same-heir-chain s (*h′*, *v′*) (*h-two*, *v-two*) ⟹ *False*
⟨*proof*⟩

lemma *one-validator-change-leaves-one-set* :
heir-after-n-switching n s (*h*, *v*) (*h′*, *v′*) ⟹
n ≤ *Suc 0* ⟹
n = *0* ∧ *FwdValidators s* (*fst* (*h*, *v*)) = *FwdValidators s* (*fst* (*h′*, *v′*)) ∨
n = *1* ∧ *FwdValidators s* (*fst* (*h*, *v*)) = *RearValidators s* (*fst* (*h′*, *v′*))
⟨*proof*⟩

lemma *prepared-by-fwd-of-origin* :

$n \leq Suc\ 0 \Longrightarrow$
$heir\text{-}after\text{-}n\text{-}switching\ n\ s\ (h,\ v)\ (h',\ v') \Longrightarrow$
$inherit\text{-}normal\ s\ (h',\ v')\ (h'',\ v'') \Longrightarrow$
$prepared\ s\ (FwdValidators\ s\ h)\ h''\ v''\ v'$

⟨*proof*⟩

lemma *heir-found-switching* :
$heir\text{-}after\text{-}n\text{-}switching\ n\ s\ (h,\ v)\ (h',\ v') \Longrightarrow$
$inherit\text{-}switching\text{-}validators\ s\ (h',\ v')\ (h'',\ v'') \Longrightarrow$
$0 < Suc\ n \Longrightarrow$
$\exists\, h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$heir\text{-}after\text{-}n\text{-}switching\ (Suc\ n - 1)\ s\ (h,\ v)\ (h\text{-}one,\ v\text{-}one) \wedge$
$inherit\text{-}switching\text{-}validators\ s\ (h\text{-}one,\ v\text{-}one)\ (h\text{-}two,\ v\text{-}two) \wedge$
$heir\text{-}after\text{-}n\text{-}switching\ 0\ s\ (h\text{-}two,\ v\text{-}two)\ (h'',\ v'')$
⟨*proof*⟩

lemma *heir-trans* :
$heir\ s\ (h\text{-}r,\ v\text{-}r)\ (h',\ v') \Longrightarrow$
$heir\ s\ (h,\ v)\ (h\text{-}r,\ v\text{-}r) \Longrightarrow$
$heir\ s\ (h,\ v)\ (h',\ v')$
⟨*proof*⟩

lemma *heir-normal-extend* :
$(\exists\, h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$heir\text{-}after\text{-}n\text{-}switching\ n\ s\ (h,\ v)\ (h\text{-}one,\ v\text{-}one) \wedge$
$inherit\text{-}switching\text{-}validators\ s\ (h\text{-}one,\ v\text{-}one)\ (h\text{-}two,\ v\text{-}two) \wedge$
$heir\text{-}after\text{-}n\text{-}switching\ 0\ s\ (h\text{-}two,\ v\text{-}two)\ (h',\ v')) \Longrightarrow$
$inherit\text{-}normal\ s\ (h',\ v')\ (h'',\ v'') \Longrightarrow$
$(\exists\, h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$heir\text{-}after\text{-}n\text{-}switching\ n\ s\ (h,\ v)\ (h\text{-}one,\ v\text{-}one) \wedge$
$inherit\text{-}switching\text{-}validators\ s\ (h\text{-}one,\ v\text{-}one)\ (h\text{-}two,\ v\text{-}two) \wedge$
$heir\text{-}after\text{-}n\text{-}switching\ 0\ s\ (h\text{-}two,\ v\text{-}two)\ (h'',\ v''))$
⟨*proof*⟩

lemma *heir-after-one-or-more-switching-dest* :
$heir\text{-}after\text{-}n\text{-}switching\ na\ s\ (h,\ v)\ (h\text{-}three,\ v\text{-}three) \Longrightarrow$
$na > 0 \Longrightarrow$
$(\exists\ h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$heir\text{-}after\text{-}n\text{-}switching\ (na - 1)\ s\ (h,\ v)\ (h\text{-}one,\ v\text{-}one) \wedge$
$inherit\text{-}switching\text{-}validators\ s\ (h\text{-}one,\ v\text{-}one)\ (h\text{-}two,\ v\text{-}two) \wedge$
$heir\text{-}after\text{-}n\text{-}switching\ 0\ s\ (h\text{-}two,\ v\text{-}two)\ (h\text{-}three,\ v\text{-}three))$
⟨*proof*⟩

lemma *high-point-still-high* :

$1 \leq n\text{-}one\text{-}pre \Longrightarrow$
$\forall\, h'\ v'.\ v < v' \longrightarrow \neg\ fork\text{-}with\text{-}center\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v')\ (h\text{-}one,\ v\text{-}one)$
$(h\text{-}two,\ v\text{-}two) \Longrightarrow$

$\neg$ *on-same-heir-chain s* (*h-one*, *v-one*) (*h-two*, *v-two*) $\Longrightarrow$
heir s (*h-orig*, *v-orig*) (*h*, *v*) $\Longrightarrow$
heir-after-n-switching n-two s (*h*, *v*) (*h-two*, *v-two*) $\Longrightarrow$
committed-by-both s h v $\Longrightarrow$
committed-by-both s h-one v-one $\Longrightarrow$
committed-by-both s h-two v-two $\Longrightarrow$
heir-after-n-switching (*Suc n-one-pre* $-$ *1*) *s* (*h*, *v*) (*h-onea*, *v-onea*) $\Longrightarrow$
inherit-switching-validators s (*h-onea*, *v-onea*) (*h-twoa*, *v-twoa*) $\Longrightarrow$
heir-after-n-switching 0 s (*h-twoa*, *v-twoa*) (*h-one*, *v-one*) $\Longrightarrow$
$\forall$ *h′ v′*. *v* $<$ *v′* $\longrightarrow$ $\neg$ *fork-with-center s* (*h-orig*, *v-orig*) (*h′*, *v′*) (*h-onea*, *v-onea*) (*h-two*, *v-two*)
⟨*proof*⟩

lemma *at-least-one-switching-means-higher* :
heir-after-n-switching n-one-pre s (*h*, *v*) (*h-onea*, *v-onea*) $\Longrightarrow$
Suc 0 $\leq$ *n-one-pre* $\Longrightarrow$
snd (*h*, *v*) $<$ *snd* (*h-onea*, *v-onea*)
⟨*proof*⟩

lemma *shallower-fork* :
heir s (*h-orig*, *v-orig*) (*h*, *v*) $\Longrightarrow$
heir-after-n-switching n-two s (*h*, *v*) (*h-two*, *v-two*) $\Longrightarrow$
committed-by-both s h v $\Longrightarrow$
committed-by-both s h-one v-one $\Longrightarrow$
committed-by-both s h-two v-two $\Longrightarrow$
heir-after-n-switching (*Suc n-one-pre* $-$ *1*) *s* (*h*, *v*) (*h-onea*, *v-onea*) $\Longrightarrow$
inherit-switching-validators s (*h-onea*, *v-onea*) (*h-twoa*, *v-twoa*) $\Longrightarrow$
heir-after-n-switching 0 s (*h-twoa*, *v-twoa*) (*h-one*, *v-one*) $\Longrightarrow$
$\neg$ *heir s* (*h-two*, *v-two*) (*h-one*, *v-one*) $\Longrightarrow$
$\neg$ *heir s* (*h-one*, *v-one*) (*h-two*, *v-two*) $\Longrightarrow$
heir s (*h-onea*, *v-onea*) (*h-two*, *v-two*) $\Longrightarrow$
v $<$ *v-onea* $\Longrightarrow$ *fork-with-center s* (*h-orig*, *v-orig*) (*h-onea*, *v-onea*) (*h-one*, *v-one*) (*h-two*, *v-two*)
⟨*proof*⟩

lemma *on-same-heir-chain-sym* :
on-same-heir-chain s (*h-one*, *v-one*) (*h-two*, *v-two*) =
on-same-heir-chain s (*h-two*, *v-two*) (*h-one*, *v-one*)
⟨*proof*⟩

lemma *fork-with-center-with-high-root-with-n-switching-sym* :
fork-with-center-with-high-root-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) *n-one* (*h-one*, *v-one*)
n-two (*h-two*, *v-two*) $\Longrightarrow$
fork-with-center-with-high-root-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) *n-two* (*h-two*, *v-two*)
n-one (*h-one*, *v-one*)
⟨*proof*⟩

2.3 Slashing Related

lemma *slashed-four-means-slashed-on-a-group*:
$finite\ X \Longrightarrow one\text{-}third\ X\ (slashed\text{-}four\ s) \Longrightarrow one\text{-}third\ X\ (slashed\ s)$
$\langle proof \rangle$

lemma *slashed-four-on-a-group*:
$finite\ (FwdValidators\ s\ h) \Longrightarrow$
$prepared\ s\ (FwdValidators\ s\ h)\ h''\ v''\ v' \Longrightarrow$
$\exists\, v\text{-}two\text{-}src.\ prepared\ s\ (FwdValidators\ s\ h)\ h\text{-}two\ v''\ v\text{-}two\text{-}src \Longrightarrow h'' \neq h\text{-}two \Longrightarrow$
$one\text{-}third\ (FwdValidators\ s\ h)\ (slashed\text{-}four\ s)$
$\langle proof \rangle$

lemma *committed-so-prepared* :
$finite\ (FwdValidators\ s\ h) \Longrightarrow$
$n\text{-}two \leq Suc\ 0 \Longrightarrow$
$heir\text{-}after\text{-}n\text{-}switching\ n\text{-}two\ s\ (h,\ v)\ (h\text{-}two,\ v'') \Longrightarrow$
$committed\text{-}by\text{-}both\ s\ h\text{-}two\ v'' \Longrightarrow$
$\neg\ one\text{-}third\ (FwdValidators\ s\ h)\ (slashed\ s) \Longrightarrow prepared\ s\ (FwdValidators\ s\ h)\ h''\ v''\ v' \Longrightarrow \exists\, v\text{-}two\text{-}src.\ prepared\ s\ (FwdValidators\ s\ h)\ h\text{-}two\ v''\ v\text{-}two\text{-}src$
$\langle proof \rangle$

lemma *slashed-three-on-a-group* :
$finite\ X \Longrightarrow$
$one\text{-}third\ X\ (\lambda n.\ (n,\ Prepare\ (h'',\ v'',\ v')) \in Messages\ s \wedge (n,\ Commit\ (h\text{-}two,\ v\text{-}two)) \in Messages\ s) \Longrightarrow$
$v' < v\text{-}two \Longrightarrow v\text{-}two < v'' \Longrightarrow one\text{-}third\ X\ (slashed\text{-}three\ s)$
$\langle proof \rangle$

lemma *slashed-three-on-group*:
$finite\ (FwdValidators\ s\ (fst\ (h,\ v))) \Longrightarrow$
$one\text{-}third\ (FwdValidators\ s\ h)\ (\lambda n.\ (n,\ Prepare\ (h'',\ v'',\ v')) \in Messages\ s \wedge (n,\ Commit\ (h\text{-}two,\ v\text{-}two)) \in Messages\ s) \Longrightarrow$
$v' < v\text{-}two \Longrightarrow$
$v\text{-}two < v'' \Longrightarrow$
$one\text{-}third\ (FwdValidators\ s\ h)\ (slashed\text{-}three\ s)$
$\langle proof \rangle$

lemma *smaller-induction-same-height-violation* :
$heir\text{-}after\text{-}n\text{-}switching\ n\ s\ (h,\ v)\ (h',\ v') \Longrightarrow$
$finite\ (FwdValidators\ s\ h) \Longrightarrow$
$prepared\text{-}by\text{-}both\ s\ h''\ v''\ v' \wedge$
$(\exists\, v\text{-}ss.\ prepared\text{-}by\text{-}both\ s\ h'\ v'\ v\text{-}ss) \wedge -\ 1 \leq v' \wedge v' < v'' \wedge nth\text{-}ancestor\ s\ (nat\ (v'' - v'))\ h'' = Some\ h' \wedge validators\text{-}match\ s\ h'\ h'' \Longrightarrow$
$n \leq Suc\ 0 \Longrightarrow$
$n\text{-}two \leq Suc\ 0 \Longrightarrow$
$\neg\ on\text{-}same\text{-}heir\text{-}chain\ s\ (h'',\ v'')\ (h\text{-}two,\ v'') \Longrightarrow$

$heir\text{-}after\text{-}n\text{-}switching\ n\text{-}two\ s\ (h,\ v)\ (h\text{-}two,\ v'') \Longrightarrow$
$committed\text{-}by\text{-}both\ s\ h\ v \Longrightarrow$
$committed\text{-}by\text{-}both\ s\ h\text{-}two\ v'' \Longrightarrow \neg\ one\text{-}third\ (FwdValidators\ s\ h)\ (slashed\ s)$
$\Longrightarrow$
$prepared\ s\ (FwdValidators\ s\ h)\ h''\ v''\ v' \Longrightarrow\ False$
⟨*proof*⟩

lemma *smaller-induction-skipping-violation* :
$heir\text{-}after\text{-}n\text{-}switching\ n\ s\ (h,\ v)\ (h',\ v') \Longrightarrow$
$finite\ (FwdValidators\ s\ h) \Longrightarrow$
$prepared\text{-}by\text{-}both\ s\ h''\ v''\ v' \land (\exists\, v\text{-}ss.\ prepared\text{-}by\text{-}both\ s\ h'\ v'\ v\text{-}ss) \land -\ 1 \le v'$
$\land\ nth\text{-}ancestor\ s\ (nat\ (v'' - v'))\ h'' = Some\ h' \land validators\text{-}match\ s\ h'\ h'' \Longrightarrow$
$v\text{-}two \le v'' \Longrightarrow$
$n \le Suc\ 0 \Longrightarrow$
$n\text{-}two \le Suc\ 0 \Longrightarrow$
$\neg\ on\text{-}same\text{-}heir\text{-}chain\ s\ (h'',\ v'')\ (h\text{-}two,\ v\text{-}two) \Longrightarrow$
$heir\text{-}after\text{-}n\text{-}switching\ n\text{-}two\ s\ (h,\ v)\ (h\text{-}two,\ v\text{-}two) \Longrightarrow$
$committed\text{-}by\text{-}both\ s\ h\ v \Longrightarrow$
$committed\text{-}by\text{-}both\ s\ h\text{-}two\ v\text{-}two \Longrightarrow$
$\neg\ one\text{-}third\ (FwdValidators\ s\ h)\ (slashed\ s) \Longrightarrow \neg\ v\text{-}two \le v' \Longrightarrow prepared\ s$
$(FwdValidators\ s\ h)\ h''\ v''\ v' \Longrightarrow$
$v\text{-}two \ne v'' \Longrightarrow False$
⟨*proof*⟩

lemma *smaller-induction-case-normal*:
$heir\text{-}after\text{-}n\text{-}switching\ n\ s\ (h,\ v)\ (h',\ v') \Longrightarrow$
$(finite\ (FwdValidators\ s\ (fst\ (h,\ v))) \Longrightarrow$
$v\text{-}two \le snd\ (h',\ v') \Longrightarrow$
$n \le Suc\ 0 \Longrightarrow$
$n\text{-}two \le Suc\ 0 \Longrightarrow$
$\neg\ on\text{-}same\text{-}heir\text{-}chain\ s\ (h',\ v')\ (h\text{-}two,\ v\text{-}two) \Longrightarrow$
$heir\text{-}after\text{-}n\text{-}switching\ n\text{-}two\ s\ (h,\ v)\ (h\text{-}two,\ v\text{-}two) \Longrightarrow$
$committed\text{-}by\text{-}both\ s\ (fst\ (h,\ v))\ (snd\ (h,\ v)) \Longrightarrow committed\text{-}by\text{-}both\ s\ h\text{-}two$
$v\text{-}two \Longrightarrow \neg\ one\text{-}third\ (FwdValidators\ s\ (fst\ (h,\ v)))\ (slashed\ s) \Longrightarrow False) \Longrightarrow$
$inherit\text{-}normal\ s\ (h',\ v')\ (h'',\ v'') \Longrightarrow$
$finite\ (FwdValidators\ s\ (fst\ (h,\ v))) \Longrightarrow$
$v\text{-}two \le snd\ (h'',\ v'') \Longrightarrow$
$n \le Suc\ 0 \Longrightarrow$
$n\text{-}two \le Suc\ 0 \Longrightarrow$
$\neg\ on\text{-}same\text{-}heir\text{-}chain\ s\ (h'',\ v'')\ (h\text{-}two,\ v\text{-}two) \Longrightarrow$
$heir\text{-}after\text{-}n\text{-}switching\ n\text{-}two\ s\ (h,\ v)\ (h\text{-}two,\ v\text{-}two) \Longrightarrow$
$committed\text{-}by\text{-}both\ s\ (fst\ (h,\ v))\ (snd\ (h,\ v)) \Longrightarrow committed\text{-}by\text{-}both\ s\ h\text{-}two\ v\text{-}two$
$\Longrightarrow \neg\ one\text{-}third\ (FwdValidators\ s\ (fst\ (h,\ v)))\ (slashed\ s) \Longrightarrow False$
⟨*proof*⟩

lemma *some-h* :
$heir\text{-}after\text{-}n\text{-}switching\ n\ s\ (h,\ v)\ (h',\ v') \Longrightarrow$
$inherit\text{-}switching\text{-}validators\ s\ (h',\ v')\ (h'',\ v'') \Longrightarrow$
$heir\ s\ (h',\ v')\ (h'',\ v'')$

$\langle proof \rangle$

lemma *smaller-induction-switching-case*:
heir-after-n-switching n s (h, v) $(h', v') \Longrightarrow$
$($*finite* $($*FwdValidators s* $($*fst* $(h, v)))) \Longrightarrow$
v-two $\leq$ *snd* $(h', v') \Longrightarrow$
$n \leq$ *Suc 0* $\Longrightarrow$
n-two $\leq$ *Suc 0* $\Longrightarrow$
$\neg$ *on-same-heir-chain s* (h', v') $($*h-two*, *v-two*$) \Longrightarrow$
heir-after-n-switching n-two s (h, v) $($*h-two*, *v-two*$) \Longrightarrow$
committed-by-both s $($*fst* $(h, v))$ $($*snd* $(h, v)) \Longrightarrow$ *committed-by-both s h-two v-two* $\Longrightarrow \neg$ *one-third* $($*FwdValidators s* $($*fst* $(h, v)))$ $($*slashed s*$) \Longrightarrow$ *False*$) \Longrightarrow$
inherit-switching-validators s (h', v') $(h'', v'') \Longrightarrow$
finite $($*FwdValidators s* $($*fst* $(h, v))) \Longrightarrow$
v-two $\leq$ *snd* $(h'', v'') \Longrightarrow$
Suc n $\leq$ *Suc 0* $\Longrightarrow$
n-two $\leq$ *Suc 0* $\Longrightarrow$
$\neg$ *on-same-heir-chain s* (h'', v'') $($*h-two*, *v-two*$) \Longrightarrow$
heir-after-n-switching n-two s (h, v) $($*h-two*, *v-two*$) \Longrightarrow$
committed-by-both s $($*fst* $(h, v))$ $($*snd* $(h, v)) \Longrightarrow$ *committed-by-both s h-two v-two* $\Longrightarrow \neg$ *one-third* $($*FwdValidators s* $($*fst* $(h, v)))$ $($*slashed s*$) \Longrightarrow$ *False*
$\langle proof \rangle$

lemma *accountable-safety-smaller-induction*:
heir-after-n-switching n-one s (h, v) $($*h-one*, *v-one*$) \Longrightarrow$
finite $($*FwdValidators s* $($*fst* $(h, v))) \Longrightarrow$
v-two $\leq$ *snd* $($*h-one*, *v-one*$) \Longrightarrow$
n-one $\leq$ *Suc 0* $\Longrightarrow$
n-two $\leq$ *Suc 0* $\Longrightarrow$
$\neg$ *on-same-heir-chain s* $($*h-one*, *v-one*$)$ $($*h-two*, *v-two*$) \Longrightarrow$
heir-after-n-switching n-two s (h, v) $($*h-two*, *v-two*$) \Longrightarrow$
committed-by-both s $($*fst* $(h, v))$ $($*snd* $(h, v))$ $(*$ *maybe not necessary* $*) \Longrightarrow$
committed-by-both s h-two v-two $\Longrightarrow$
$\neg$ *one-third* $($*FwdValidators s* $($*fst* $(h, v)))$ $($*slashed s*$) \Longrightarrow$ *False*
$\langle proof \rangle$

lemma *accountable-safety-from-fork-with-high-root-base-one-longer* :
n-one $\leq$ *1* $\wedge$
n-two $\leq$ *1* $\wedge$
v-one $\geq$ *v-two* $\Longrightarrow$
finite $($*FwdValidators s h*$) \Longrightarrow$
fork-with-center-with-high-root-with-n-switching
s $($*h-orig*, *v-orig*$)$ (h, v) *n-one* $($*h-one*, *v-one*$)$ *n-two* $($*h-two*, *v-two*$) \Longrightarrow$
$\exists\ h'\ v'.$
heir s $($*h-orig*, *v-orig*$)$ $(h', v') \wedge$
one-third-of-fwd-slashed s h'
$\langle proof \rangle$

lemma *accountable-safety-from-fork-with-high-root-base-two-longer* :
n-one $\leq$ *1* $\wedge$
n-two $\leq$ *1* $\wedge$
v-one $\leq$ *v-two* $\Longrightarrow$
finite (*FwdValidators s h*) $\Longrightarrow$
fork-with-center-with-high-root-with-n-switching
s (*h-orig*, *v-orig*) (*h*, *v*) *n-one* (*h-one*, *v-one*) *n-two* (*h-two*, *v-two*) $\Longrightarrow$
$\exists$ *h′ v′*.
heir s (*h-orig*, *v-orig*) (*h′*, *v′*) $\wedge$
one-third-of-fwd-slashed s h′
⟨*proof*⟩

lemma *accountable-safety-from-fork-with-high-root-base* :
n-one $\leq$ *1* $\wedge$
n-two $\leq$ *1* $\wedge$
fork-with-center-with-high-root-with-n-switching
s (*h-orig*, *v-orig*) (*h*, *v*) *n-one* (*h-one*, *v-one*) *n-two* (*h-two*, *v-two*) $\Longrightarrow$
finite (*FwdValidators s h*) $\Longrightarrow$
$\exists$ *h′ v′*.
heir s (*h-orig*, *v-orig*) (*h′*, *v′*) $\wedge$
one-third-of-fwd-slashed s h′

⟨*proof*⟩

2.4 Mainline Arguments for Accountable Safety

lemma *use-highness* :
1 $\leq$ *n-one-pre* $\Longrightarrow$
$\forall$ *h′ v′*. *v* < *v′* $\longrightarrow$ $\neg$ *fork-with-center s* (*h-orig*, *v-orig*) (*h′*, *v′*) (*h-one*, *v-one*) (*h-two*, *v-two*) $\Longrightarrow$
heir s (*h-orig*, *v-orig*) (*h*, *v*) $\Longrightarrow$
heir-after-n-switching n-two s (*h*, *v*) (*h-two*, *v-two*) $\Longrightarrow$
committed-by-both s h v $\Longrightarrow$
committed-by-both s h-one v-one $\Longrightarrow$
committed-by-both s h-two v-two $\Longrightarrow$
heir-after-n-switching (*Suc n-one-pre* $-$ *1*) *s* (*h*, *v*) (*h-onea*, *v-onea*) $\Longrightarrow$
inherit-switching-validators s (*h-onea*, *v-onea*) (*h-twoa*, *v-twoa*) $\Longrightarrow$
heir-after-n-switching 0 s (*h-twoa*, *v-twoa*) (*h-one*, *v-one*) $\Longrightarrow$
$\neg$ *heir s* (*h-two*, *v-two*) (*h-one*, *v-one*) $\Longrightarrow$
$\neg$ *heir s* (*h-one*, *v-one*) (*h-two*, *v-two*) $\Longrightarrow$ *heir s* (*h-onea*, *v-onea*) (*h-two*, *v-two*) $\Longrightarrow$ *False*
⟨*proof*⟩

lemma *confluence-should-not*:
1 $\leq$ *n-one-pre* $\Longrightarrow$
$\forall$ *h′ v′*. *v* < *v′* $\longrightarrow$ $\neg$ *fork-with-center s* (*h-orig*, *v-orig*) (*h′*, *v′*) (*h-one*, *v-one*) (*h-two*, *v-two*) $\Longrightarrow$
heir s (*h-orig*, *v-orig*) (*h*, *v*) $\Longrightarrow$

heir-after-n-switching n-two s (*h*, *v*) (*h-two*, *v-two*) $\Longrightarrow$
committed-by-both s h v $\Longrightarrow$
committed-by-both s h-one v-one $\Longrightarrow$
committed-by-both s h-two v-two $\Longrightarrow$
heir-after-n-switching (*Suc n-one-pre* − *1*) *s* (*h*, *v*) (*h-onea*, *v-onea*) $\Longrightarrow$
inherit-switching-validators s (*h-onea*, *v-onea*) (*h-twoa*, *v-twoa*) $\Longrightarrow$
heir-after-n-switching 0 s (*h-twoa*, *v-twoa*) (*h-one*, *v-one*) $\Longrightarrow$
$\neg$ *heir s* (*h-two*, *v-two*) (*h-one*, *v-one*) $\Longrightarrow$
$\neg$ *heir s* (*h-one*, *v-one*) (*h-two*, *v-two*) $\Longrightarrow$ *heir s* (*h-two*, *v-two*) (*h-onea*, *v-onea*) $\Longrightarrow$ *False*
⟨*proof*⟩

lemma *prev-switch-not-on-same-heir-chain* :
1 $\leq$ *n-one-pre* $\Longrightarrow$
$\forall$ *h′ v′*. *v* < *v′* $\longrightarrow$ $\neg$ *fork-with-center s* (*h-orig*, *v-orig*) (*h′*, *v′*) (*h-one*, *v-one*) (*h-two*, *v-two*) $\Longrightarrow$
$\neg$ *on-same-heir-chain s* (*h-one*, *v-one*) (*h-two*, *v-two*) $\Longrightarrow$
heir s (*h-orig*, *v-orig*) (*h*, *v*) $\Longrightarrow$
heir-after-n-switching n-two s (*h*, *v*) (*h-two*, *v-two*) $\Longrightarrow$
committed-by-both s h v $\Longrightarrow$
committed-by-both s h-one v-one $\Longrightarrow$
committed-by-both s h-two v-two $\Longrightarrow$
heir-after-n-switching (*Suc n-one-pre* − *1*) *s* (*h*, *v*) (*h-onea*, *v-onea*) $\Longrightarrow$
inherit-switching-validators s (*h-onea*, *v-onea*) (*h-twoa*, *v-twoa*) $\Longrightarrow$
heir-after-n-switching 0 s (*h-twoa*, *v-twoa*) (*h-one*, *v-one*) $\Longrightarrow$
$\neg$ *on-same-heir-chain s* (*h-onea*, *v-onea*) (*h-two*, *v-two*)
⟨*proof*⟩

lemma *reduce-fork* :
fork-with-center-with-high-root-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) (*Suc n-one-pre*) (*h-one*, *v-one*)
n-two (*h-two*, *v-two*) $\Longrightarrow$
1 $\leq$ *n-one-pre* $\Longrightarrow$
$\exists$ *h-one′ v-one′*.
fork-with-center-with-high-root-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) *n-one-pre* (*h-one′*, *v-one′*)
n-two (*h-two*, *v-two*)
⟨*proof*⟩

lemma *switching-induction-case-one* :
$\forall$ *n-one n-twoa h-one v-one h-two v-two*.
n-one + *n-twoa* $\leq$ *n-one-pre* + *n-two* $\longrightarrow$
finite (*FwdValidators s h*) $\longrightarrow$
fork-with-center-with-high-root-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) *n-one* (*h-one*, *v-one*) *n-twoa*
(*h-two*, *v-two*) $\longrightarrow$
($\exists$ *h′ v′*. *heir s* (*h-orig*, *v-orig*) (*h′*, *v′*) $\wedge$ *one-third-of-fwd-slashed s h′*) $\Longrightarrow$
finite (*FwdValidators s h*) $\Longrightarrow$
fork-with-center-with-high-root-with-n-switching s (*h-orig*, *v-orig*) (*h*, *v*) (*Suc*

$n\text{-}one\text{-}pre)$ $(h\text{-}one,\ v\text{-}one)$
$n\text{-}two$ $(h\text{-}two,\ v\text{-}two) \Longrightarrow$
$1 \le n\text{-}one\text{-}pre \Longrightarrow$
$k = n\text{-}one\text{-}pre + n\text{-}two \Longrightarrow$
$\exists\, h'\ v'.\ heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h'$
$\langle proof \rangle$

lemma *some-symmetry* :
$\forall\, n\text{-}onea\ n\text{-}two\ h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$n\text{-}onea + n\text{-}two \le n\text{-}one + n\text{-}two\text{-}pre \longrightarrow$
$finite\ (FwdValidators\ s\ h) \longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\text{-}with\text{-}n\text{-}switching\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)\ n\text{-}onea$
$(h\text{-}one,\ v\text{-}one)\ n\text{-}two$
$(h\text{-}two,\ v\text{-}two) \longrightarrow$
$(\exists\, h'\ v'.\ heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h') \Longrightarrow$
$\forall\, n\text{-}onea\ n\text{-}twoa\ h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$n\text{-}onea + n\text{-}twoa \le n\text{-}two\text{-}pre + n\text{-}one \longrightarrow$
$finite\ (FwdValidators\ s\ h) \longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\text{-}with\text{-}n\text{-}switching\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)\ n\text{-}onea$
$(h\text{-}one,\ v\text{-}one)\ n\text{-}twoa$
$(h\text{-}two,\ v\text{-}two) \longrightarrow$
$(\exists\, h'\ v'.\ heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h')$
$\langle proof \rangle$

lemma *switching-induction-case-two* :
$\forall\, n\text{-}onea\ n\text{-}two\ h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$n\text{-}onea + n\text{-}two \le n\text{-}one + n\text{-}two\text{-}pre \longrightarrow$
$finite\ (FwdValidators\ s\ h) \longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\text{-}with\text{-}n\text{-}switching\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)$
$n\text{-}onea\ (h\text{-}one,\ v\text{-}one)\ n\text{-}two$
$(h\text{-}two,\ v\text{-}two) \longrightarrow$
$(\exists\, h'\ v'.\ heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h') \Longrightarrow$
$finite\ (FwdValidators\ s\ h) \Longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\text{-}with\text{-}n\text{-}switching\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)\ n\text{-}one$
$(h\text{-}one,\ v\text{-}one)$
$(Suc\ n\text{-}two\text{-}pre)\ (h\text{-}two,\ v\text{-}two) \Longrightarrow$
$1 \le n\text{-}two\text{-}pre \Longrightarrow$
$k = n\text{-}one + n\text{-}two\text{-}pre \Longrightarrow$
$\exists\, h'\ v'.\ heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h'$
$\langle proof \rangle$

lemma *switching-induction* :
$\forall\, n\text{-}one\ n\text{-}two\ h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$n\text{-}one + n\text{-}two \le k \longrightarrow$
$finite\ (FwdValidators\ s\ h) \longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\text{-}with\text{-}n\text{-}switching\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)$
$n\text{-}one\ (h\text{-}one,\ v\text{-}one)\ n\text{-}two$
$(h\text{-}two,\ v\text{-}two) \longrightarrow$

$(\exists h'\ v'.\ heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h')$
$\Longrightarrow$
$\forall n\text{-}one\ n\text{-}two\ h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$n\text{-}one + n\text{-}two \leq Suc\ k \longrightarrow$
$finite\ (FwdValidators\ s\ h) \longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\text{-}with\text{-}n\text{-}switching\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)$
$n\text{-}one\ (h\text{-}one,\ v\text{-}one)\ n\text{-}two$
$(h\text{-}two,\ v\text{-}two) \longrightarrow$
$(\exists h'\ v'.\ heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h')$
$\langle proof \rangle$

lemma *accountable-safety-from-fork-with-high-root-with-n-ind* :
$\forall\ n\text{-}one\ n\text{-}two\ h\text{-}one\ v\text{-}one\ h\text{-}two\ v\text{-}two.$
$n\text{-}one + n\text{-}two \leq k \longrightarrow$
$finite\ (FwdValidators\ s\ h) \longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\text{-}with\text{-}n\text{-}switching$
$s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)\ n\text{-}one\ (h\text{-}one,\ v\text{-}one)\ n\text{-}two\ (h\text{-}two,\ v\text{-}two) \longrightarrow$
$(\exists\ h'\ v'.$
$heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge$
$one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h')$
$\langle proof \rangle$

lemma *accountable-safety-from-fork-with-high-root-with-n* :
$finite\ (FwdValidators\ s\ h) \Longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\text{-}with\text{-}n\text{-}switching$
$s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)\ n\text{-}one\ (h\text{-}one,\ v\text{-}one)\ n\text{-}two\ (h\text{-}two,\ v\text{-}two) \Longrightarrow$
$\exists\ h'\ v'.$
$heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge$
$one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h'$
$\langle proof \rangle$

lemma *accountable-safety-from-fork-with-high-root* :
$finite\ (FwdValidators\ s\ h) \Longrightarrow$
$fork\text{-}with\text{-}center\text{-}with\text{-}high\text{-}root\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h,\ v)\ (h\text{-}one,\ v\text{-}one)\ (h\text{-}two,$
$v\text{-}two) \Longrightarrow$
$\exists\ h'\ v'.$
$heir\ s\ (h\text{-}orig,\ v\text{-}orig)\ (h',\ v') \wedge$
$one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h'$
$\langle proof \rangle$

definition *validator-sets-finite* :: *situation* $\Rightarrow$ *bool*
where $validator\text{-}sets\text{-}finite\ s = (\forall\ h.\ finite\ (FwdValidators\ s\ h))$

lemma *accountable-safety-center* :
$validator\text{-}sets\text{-}finite\ s \Longrightarrow$
$fork\text{-}with\text{-}center\ s\ (h,\ v)\ (h,\ v)\ (h1,\ v1)\ (h2,\ v2) \Longrightarrow$
$\exists\ h'\ v'.$
$heir\ s\ (h,\ v)\ (h',\ v') \wedge$
$one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h'$

⟨*proof*⟩

lemma *heir-initial* :
$heir\ s\ (h,\ v)\ (h1,\ v1) \Longrightarrow$
$heir\ s\ (h,\ v)\ (h,\ v)$
⟨*proof*⟩

lemma *fork-with-center-and-root* :
$fork\text{-}with\text{-}commits\ s\ (h,\ v)\ (h1,\ v1)\ (h2,\ v2) \Longrightarrow$
$fork\text{-}with\text{-}center\ s\ (h,\ v)\ (h,\ v)\ (h1,\ v1)\ (h2,\ v2)$

⟨*proof*⟩

3 Accountable Safety (don't skip)

The statement of accountable safety is simple. If a situation has a finite number of validators on each hash, a fork means some validator set suffers 1/3 slashing. A fork is defined using the *heir* relation. The slashed validator set is also a heir of the original validator set.

lemma *accountable-safety* :
$validator\text{-}sets\text{-}finite\ s \Longrightarrow$
$fork\text{-}with\text{-}commits\ s\ (h,\ v)\ (h1,\ v1)\ (h2,\ v2) \Longrightarrow$
$\exists\ h'\ v'.$
$heir\ s\ (h,\ v)\ (h',\ v') \land$
$one\text{-}third\text{-}of\text{-}fwd\text{-}slashed\ s\ h'$
⟨*proof*⟩

end