Minimal CBC Casper Isabelle/HOL proofs

LayerX

February 21, 2019

Contents

1 Description of CBC Casper

theory *CBCCasper*

imports *Main HOL.Real Libraries/Strict-Order Libraries/Restricted-Predicates Libraries/LaTeXsugar*

begin

notation *Set.empty* (∅)

typedecl *validator*

typedecl *consensus-value*

datatype *message* =
  Message consensus-value ∗ *validator* ∗ *message list*

type-synonym *state* = *message set*

fun *sender* :: *message* $\Rightarrow$ *validator*
where
sender (*Message* (-, *v*, -)) = *v*

fun *est* :: *message* $\Rightarrow$ *consensus-value*
where
est (*Message* (*c*, -, -)) = *c*

fun *justification* :: *message* $\Rightarrow$ *state*
where
justification (*Message* (-, -, *s*)) = *set s*

fun
Σ-*i* :: (*validator set* $\times$ *consensus-value set* $\times$ (*message set* $\Rightarrow$ *consensus-value set*)) $\Rightarrow$ *nat* $\Rightarrow$ *state set* **and**
M-i :: (*validator set* $\times$ *consensus-value set* $\times$ (*message set* $\Rightarrow$ *consensus-value set*)) $\Rightarrow$ *nat* $\Rightarrow$ *message set*
where
Σ-*i* (V,C,ε) *0* = $\{\emptyset\}$
| Σ-*i* (V,C,ε) n = $\{\sigma \in$ *Pow* (*M-i* (V,C,ε) $(n - 1)$). *finite* $\sigma \wedge (\forall\ m.\ m \in \sigma \longrightarrow$ *justification* $m \subseteq \sigma)\}$
| *M-i* (V,C,ε) n = $\{m.$ *est* $m \in C \wedge$ *sender* $m \in V \wedge$ *justification* $m \in$ (Σ-*i* (V,C,ε) n) $\wedge$ *est* $m \in \varepsilon$ (*justification* m)$\}$

locale *Params* =
fixes V :: *validator set*
and W :: *validator* $\Rightarrow$ *real*
and t :: *real*
fixes C :: *consensus-value set*
and ε :: *message set* $\Rightarrow$ *consensus-value set*

begin
definition $\Sigma = (\bigcup i{\in}\mathbb{N}.\ \Sigma$-*i* $(V,C,\varepsilon)\ i)$
definition $M = (\bigcup i{\in}\mathbb{N}.$ *M-i* $(V,C,\varepsilon)\ i)$
definition *is-valid-estimator* :: (*state* $\Rightarrow$ *consensus-value set*) $\Rightarrow$ *bool*
where
is-valid-estimator $e = (\forall \sigma \in \Sigma.\ e\ \sigma \in$ *Pow* $C - \{\emptyset\})$

lemma Σ*i-subset-Mi*: Σ-*i* (V,C,ε) $(n + 1) \subseteq$ *Pow* (*M-i* (V,C,ε) n)
by *force*

lemma Σ*i-subset-to-Mi*: Σ-*i* (V,C,ε) $n \subseteq \Sigma$-*i* (V,C,ε) $(n+1) \Longrightarrow$ *M-i* (V,C,ε) $n \subseteq$ *M-i* (V,C,ε) $(n+1)$
by *auto*

lemma *Mi-subset-to-Σi*: *M-i* (V,C,ε) $n \subseteq$ *M-i* (V,C,ε) $(n+1) \Longrightarrow$ *Σ-i* (V,C,ε) $(n+1) \subseteq$ *Σ-i* (V,C,ε) $(n+2)$
by *auto*

lemma *Σi-monotonic*: *Σ-i* (V,C,ε) $n \subseteq$ *Σ-i* (V,C,ε) $(n+1)$
apply (*induction n*)
apply *simp*
apply (*metis Mi-subset-to-Σi Suc-eq-plus1 Σi-subset-to-Mi add.commute add-2-eq-Suc*)
done

lemma *Mi-monotonic*: *M-i* (V,C,ε) $n \subseteq$ *M-i* (V,C,ε) $(n+1)$
apply (*induction n*)
defer
using *Σi-monotonic Σi-subset-to-Mi* **apply** *blast*
apply *auto*
done

lemma *message-is-in-M-i* :
$\forall\ m \in M.\ \exists\ n \in \mathbb{N}.\ m \in$ *M-i* $(V,\ C,\ \varepsilon)\ (n - 1)$
apply (*simp add*: *M-def Σ-i.elims*)
by (*metis Nats-1 Nats-add One-nat-def diff-Suc-1 plus-1-eq-Suc*)

lemma *state-is-in-pow-M-i* :
$\forall\ \sigma \in \Sigma.\ (\exists\ n \in \mathbb{N}.\ \sigma \in$ *Pow* (*M-i* $(V,\ C,\ \varepsilon)\ (n - 1)) \land (\forall\ m \in \sigma.$ *justification* $m \subseteq \sigma))$
apply (*simp add*: *Σ-def*)

apply *auto*
proof −
fix y :: *nat* **and** σ :: *message set*
assume *a1*: $\sigma \in$ *Σ-i* $(V,\ C,\ \varepsilon)\ y$
assume *a2*: $y \in \mathbb{N}$
have $\sigma \subseteq$ *M-i* $(V,\ C,\ \varepsilon)\ y$
using *a1* **by** (*meson Params.Σi-monotonic Params.Σi-subset-Mi Pow-iff contra-subsetD*)
then have $\exists n.\ n \in \mathbb{N} \land \sigma \subseteq$ *M-i* $(V,\ C,\ \varepsilon)\ (n - 1)$
using *a2* **by** (*metis* (*no-types*) *Nats-1 Nats-add diff-Suc-1 plus-1-eq-Suc*)
then show $\exists n{\in}\mathbb{N}.\ \sigma \subseteq \{m.$ *est* $m \in C \land$ *sender* $m \in V \land$ *justification* $m \in$ *Σ-i* $(V,\ C,\ \varepsilon)\ (n -$ *Suc 0*$) \land$ *est* $m \in \varepsilon$ (*justification* m)$\}$
by *auto*
next
show $\bigwedge y\ \sigma\ m\ x.\ y \in \mathbb{N} \Longrightarrow \sigma \in$ *Σ-i* $(V,\ C,\ \varepsilon)\ y \Longrightarrow m \in \sigma \Longrightarrow x \in$ *justification* $m \Longrightarrow x \in \sigma$
using *Params.Σi-monotonic* **by** *fastforce*
qed

lemma *message-is-in-M-i-n* :
$\forall\ m \in M.\ \exists\ n \in \mathbb{N}.\ m \in$ *M-i* $(V,\ C,\ \varepsilon)\ n$

by (*smt Mi-monotonic Suc-diff-Suc add-leE diff-add diff-le-self message-is-in-M-i neq0-conv plus-1-eq-Suc subsetCE zero-less-diff*)

lemma *message-in-state-is-valid* :
  $\forall\ \sigma\ m.\ \sigma \in \Sigma \wedge m \in \sigma \longrightarrow\ \ m \in M$
  apply (*rule*, *rule*, *rule*)
proof −
  fix $\sigma\ m$
  assume $\sigma \in \Sigma \wedge m \in \sigma$
  have
    $\exists\ n \in \mathbb{N}.\ m \in M\text{-}i\ (V,\ C,\ \varepsilon)\ n$
    $\Longrightarrow m \in M$
    using *M-def* **by** *blast*
  then show
    $m \in M$
    apply (*simp add*: *M-def*)
   by (*smt M-i.simps Params.Σi-monotonic PowD Suc-diff-Suc* ‹$\sigma \in \Sigma \wedge m \in \sigma$› *add-leE diff-add diff-le-self gr0I mem-Collect-eq plus-1-eq-Suc state-is-in-pow-M-i subsetCE zero-less-diff*)
qed

lemma *state-is-subset-of-M* : $\forall\ \sigma \in \Sigma.\ \sigma \subseteq M$
  using *message-in-state-is-valid* **by** *blast*

lemma *state-difference-is-valid-message* :
  $\forall\ \sigma\ \sigma'.\ \sigma \in \Sigma \wedge \sigma' \in \Sigma$
  $\longrightarrow$ *is-future-state*$(\sigma,\ \sigma')$
  $\longrightarrow \sigma' - \sigma \subseteq M$
  using *state-is-subset-of-M* **by** *blast*

lemma *state-is-finite* : $\forall\ \sigma \in \Sigma.$ *finite* σ
  apply (*simp add*: Σ-*def*)
  using *Params.Σi-monotonic* **by** *fastforce*

lemma *justification-is-finite* : $\forall\ m \in M.$ *finite* (*justification* m)
  apply (*simp add*: *M-def*)
  using *Params.Σi-monotonic* **by** *fastforce*

lemma Σ-*is-subseteq-of-pow-M*: $\Sigma \subseteq Pow\ M$
  by (*simp add*: *state-is-subset-of-M subsetI*)

lemma *M-type*: $\bigwedge m.\ m \in M \Longrightarrow est\ m \in C \wedge sender\ m \in V \wedge justification\ m \in \Sigma$
  unfolding *M-def* Σ-*def*
  by *auto*

end

locale *Protocol* = *Params* +
  assumes *V-type*: $V \neq \emptyset$
  and *W-type*: $\bigwedge w.\ w \in range\ W \Longrightarrow w > 0$
  and *t-type*: $0 \leq t\ t < Sum\ (W\ `\ V)$
  and *C-type*: $card\ C > 1$
  and *ε-type*: *is-valid-estimator* ε

lemma (**in** *Protocol*) *estimates-are-non-empty*: $\bigwedge\ \sigma.\ \sigma \in \Sigma \Longrightarrow \varepsilon\ \sigma \neq \emptyset$
  using *is-valid-estimator-def ε-type* **by** *auto*

lemma (**in** *Protocol*) *estimates-are-subset-of-C*: $\bigwedge\ \sigma.\ \sigma \in \Sigma \Longrightarrow \varepsilon\ \sigma \subseteq C$
  using *is-valid-estimator-def ε-type* **by** *auto*

lemma (**in** *Params*) *empty-set-exists-in-Σ-0*: $\emptyset \in \Sigma\text{-}i\ (V,\ C,\ \varepsilon)\ 0$
  by *simp*

lemma (**in** *Params*) *empty-set-exists-in-Σ*: $\emptyset \in \Sigma$
  apply (*simp add*: Σ-*def*)
  using *Nats-0* Σ-*i.simps*(*1*) **by** *blast*

lemma (**in** *Params*) Σ-*i-is-non-empty*: $\Sigma\text{-}i\ (V,\ C,\ \varepsilon)\ n \neq \emptyset$
  apply (*induction n*)
  using *empty-set-exists-in-Σ-0* **by** *auto*

lemma (**in** *Params*) Σ-*is-non-empty*: $\Sigma \neq \emptyset$
  using *empty-set-exists-in-Σ* **by** *blast*

lemma (**in** *Protocol*) *estimates-exists-for-empty-set* :
  $\varepsilon\ \emptyset \neq \emptyset$
  by (*simp add*: *empty-set-exists-in-Σ estimates-are-non-empty*)

lemma (**in** *Protocol*) *non-justifying-message-exists-in-M-0*:
  $\exists\ m.\ m \in M\text{-}i\ (V,\ C,\ \varepsilon)\ 0 \land justification\ m = \emptyset$
  apply *auto*
proof −
  have $\varepsilon\ \emptyset \subseteq C$
    using *Params.empty-set-exists-in-Σ ε-type is-valid-estimator-def* **by** *auto*
  then show $\exists m.\ est\ m \in C \land sender\ m \in V \land justification\ m = \emptyset \land est\ m \in \varepsilon\ (justification\ m) \land justification\ m = \emptyset$
    by (*metis V-type all-not-in-conv est.simps estimates-exists-for-empty-set justification.simps sender.simps set-empty subsetCE*)
qed

lemma (**in** *Protocol*) *M-i-is-non-empty*: $M\text{-}i\ (V,\ C,\ \varepsilon)\ n \neq \emptyset$
  apply (*induction n*)
  using *non-justifying-message-exists-in-M-0* **apply** *auto*
  using *Mi-monotonic empty-iff empty-subsetI* **by** *fastforce*

lemma (**in** *Protocol*) *M-is-non-empty*: $M \neq \emptyset$

using *non-justifying-message-exists-in-M-0 M-def Nats-0* **by** *blast*

lemma (**in** *Protocol*) *C-is-not-empty* : $C \neq \emptyset$
using *C-type* **by** *auto*

lemma (**in** *Params*) Σ*i-is-subset-of-*Σ :
$\forall\ n \in \mathbb{N}.\ \Sigma\text{-}i\ (V,\ C,\ \varepsilon)\ n \subseteq \Sigma$
by (*simp add*: Σ*-def SUP-upper*)

lemma (**in** *Protocol*) *message-justifying-state-in-*Σ*-n-exists-in-M-n* :
$\forall\ n \in \mathbb{N}.\ (\forall\ \sigma.\ \sigma \in \Sigma\text{-}i\ (V,\ C,\ \varepsilon)\ n \longrightarrow (\exists\ m.\ m \in M\text{-}i\ (V,\ C,\ \varepsilon)\ n \wedge$ *justification* $m = \sigma))$
apply *auto*
proof −
fix $n\ \sigma$
assume $n \in \mathbb{N}$
and $\sigma \in \Sigma\text{-}i\ (V,\ C,\ \varepsilon)\ n$
then have $\sigma \in \Sigma$
using Σ*i-is-subset-of-*Σ **by** *auto*
have $\varepsilon\ \sigma \neq \emptyset$
using *estimates-are-non-empty* $\langle\sigma \in \Sigma\rangle$ **by** *auto*
have *finite* σ
using *state-is-finite* $\langle\sigma \in \Sigma\rangle$ **by** *auto*
moreover have $\exists\ m.$ *sender* $m \in V \wedge$ *est* $m \in \varepsilon\ \sigma \wedge$ *justification* $m = \sigma$
using *est.simps sender.simps justification.simps V-type* $\langle\varepsilon\ \sigma \neq \emptyset\rangle$ $\langle$*finite* $\sigma\rangle$
by (*metis all-not-in-conv finite-list*)
moreover have $\varepsilon\ \sigma \subseteq C$
using *estimates-are-subset-of-C* Σ*i-is-subset-of-*Σ $\langle n \in \mathbb{N}\rangle$ $\langle\sigma \in \Sigma\text{-}i\ (V,\ C,\ \varepsilon)\ n\rangle$ **by** *blast*
ultimately show $\exists\ m.$ *est* $m \in C \wedge$ *sender* $m \in V \wedge$ *justification* $m \in \Sigma\text{-}i\ (V,\ C,\ \varepsilon)\ n \wedge$ *est* $m \in \varepsilon$ (*justification* m) $\wedge$ *justification* $m = \sigma$
using *Nats-1 One-nat-def*
using $\langle\sigma \in \Sigma\text{-}i\ (V,\ C,\ \varepsilon)\ n\rangle$ **by** *blast*
qed

lemma (**in** *Protocol*) Σ*-type*: $\Sigma \subset$ *Pow M*
proof −
obtain m **where** $m \in M\text{-}i\ (V,\ C,\ \varepsilon)\ 0 \wedge$ *justification* $m = \emptyset$
using *non-justifying-message-exists-in-M-0* **by** *auto*
then have $\{m\} \in \Sigma\text{-}i\ (V,\ C,\ \varepsilon)$ (*Suc 0*)
using *Params.*Σ*i-subset-Mi* **by** *auto*
then have $\exists\ m'.\ m' \in M\text{-}i\ (V,\ C,\ \varepsilon)$ (*Suc 0*) $\wedge$ *justification* $m' = \{m\}$
using *message-justifying-state-in-*Σ*-n-exists-in-M-n Nats-1 One-nat-def* **by** *metis*
then obtain m' **where** $m' \in M\text{-}i\ (V,\ C,\ \varepsilon)$ (*Suc 0*) $\wedge$ *justification* $m' = \{m\}$ **by** *auto*
then have $\{m'\} \in$ *Pow M*
using *M-def*
by (*metis Nats-1 One-nat-def PowD PowI Pow-bottom UN-I insert-subset*)

moreover have $\{m'\} \notin \Sigma$
using *Params.state-is-in-pow-M-i Protocol-axioms* ‹$m' \in$ *M-i* $(V,\ C,\ \varepsilon)$ (*Suc 0*) $\wedge$ *justification* $m' = \{m\}$› **by** *fastforce*
ultimately show *?thesis*
using Σ*-is-subseteq-of-pow-M* **by** *auto*
qed

lemma (**in** *Protocol*) *M-type-counterexample*:
$(\forall\ \sigma.\ \varepsilon\ \sigma = C) \Longrightarrow M = \{m.\ \mathit{est}\ m \in C \wedge \mathit{sender}\ m \in V \wedge \mathit{justification}\ m \in \Sigma\}$
apply (*simp add*: *M-def*)
apply *auto*
using Σ*i-is-subset-of-*Σ **apply** *blast*
by (*simp add*: Σ*-def*)

definition *observed* :: *message set* $\Rightarrow$ *validator set*
where
observed $\sigma = \{\mathit{sender}\ m \mid m.\ m \in \sigma\}$

lemma (**in** *Protocol*) *observed-type* :
$\forall\ \sigma \in \Sigma.\ \mathit{observed}\ \sigma \subseteq V$
using *Params.M-type Protocol-axioms observed-def state-is-subset-of-M* **by** *fastforce*

fun *is-future-state* :: (*state* $*$ *state*) $\Rightarrow$ *bool*
where
is-future-state $(\sigma 1,\ \sigma 2) = (\sigma 1 \subseteq \sigma 2)$

definition *justified* :: *message* $\Rightarrow$ *message* $\Rightarrow$ *bool*
where
justified m1 m2 = (*m1* $\in$ *justification m2*)

definition *equivocation* :: (*message* $*$ *message*) $\Rightarrow$ *bool*
where
equivocation =
$(\lambda(m1,\ m2).\ \mathit{sender}\ m1 = \mathit{sender}\ m2 \wedge m1 \neq m2 \wedge \neg\ (\mathit{justified}\ m1\ m2) \wedge \neg\ (\mathit{justified}\ m2\ m1))$

definition *is-equivocating* :: *state* $\Rightarrow$ *validator* $\Rightarrow$ *bool*
where
is-equivocating $\sigma\ v = (\exists\ m1 \in \sigma.\ \exists\ m2 \in \sigma.\ \mathit{equivocation}\ (m1,\ m2) \wedge \mathit{sender}\ m1 = v)$

definition *equivocating-validators* :: *state* $\Rightarrow$ *validator set*
where

equivocating-validators σ = $\{v \in$ *observed* σ. *is-equivocating* σ $v\}$

lemma (**in** *Protocol*) *equivocating-validators-type* :
$\forall$ $\sigma \in \Sigma$. *equivocating-validators* $\sigma \subseteq V$
using *observed-type equivocating-validators-def* **by** *blast*

definition (**in** *Params*) *equivocating-validators-paper* :: *state* $\Rightarrow$ *validator set*
where
equivocating-validators-paper σ = $\{v \in V$. *is-equivocating* σ $v\}$

lemma (**in** *Protocol*) *equivocating-validators-is-equivalent-to-paper* :
$\forall$ $\sigma \in \Sigma$. *equivocating-validators* σ = *equivocating-validators-paper* σ
by (*smt Collect-cong Params.equivocating-validators-paper-def equivocating-validators-def is-equivocating-def mem-Collect-eq observed-type observed-def subsetCE*)

definition (**in** *Params*) *equivocation-fault-weight* :: *state* $\Rightarrow$ *real*
where
equivocation-fault-weight σ = *sum W* (*equivocating-validators* σ)

definition (**in** *Params*) *is-faults-lt-threshold* :: *state* $\Rightarrow$ *bool*
where
is-faults-lt-threshold σ = (*equivocation-fault-weight* $\sigma < t$)

definition (**in** *Protocol*) Σt :: *state set*
where
Σt = $\{\sigma \in \Sigma$. *is-faults-lt-threshold* $\sigma\}$

lemma (**in** *Protocol*) Σt*-is-subset-of-*Σ : $\Sigma t \subseteq \Sigma$
using Σt*-def* **by** *auto*

type-synonym *state-property* = *state* $\Rightarrow$ *bool*

type-synonym *consensus-value-property* = *consensus-value* $\Rightarrow$ *bool*

definition (**in** *Params*) *message-justification* :: *message rel*
where
message-justification = $\{(m1, m2)$. $\{m1, m2\} \subseteq M \wedge$ *justified m1 m2*$\}$

lemma (**in** *Protocol*) *transitivity-of-justifications* :

trans message-justification
apply (*simp add*: *trans-def message-justification-def justified-def*)
by (*meson Params.M-type Params.state-is-in-pow-M-i Protocol-axioms contra-subsetD*)

lemma (**in** *Protocol*) *irreflexivity-of-justifications* :
irrefl message-justification
apply (*simp add*: *irrefl-def message-justification-def justified-def*)
apply (*simp add*: *M-def*)
apply *auto*
proof −
fix *n m*
assume *est m* ∈ *C*
assume *sender m* ∈ *V*
assume *justification m* ∈ Σ-*i* (*V*, *C*, *ε*) *n*
assume *est m* ∈ *ε* (*justification m*)
assume *m* ∈ *justification m*
have *m* ∈ *M-i* (*V*, *C*, *ε*) (*n* − *1*)
by (*smt M-i.simps One-nat-def Params.Σi-subset-Mi Pow-iff Suc-pred* ‹*est m* ∈ *C*› ‹*est m* ∈ *ε* (*justification m*)› ‹*justification m* ∈ Σ-*i* (*V*, *C*, *ε*) *n*› ‹*m* ∈ *justification m*› ‹*sender m* ∈ *V*› *add.right-neutral add-Suc-right diff-is-0-eq′ diff-le-self diff-zero mem-Collect-eq not-gr0 subsetCE*)
then have *justification m* ∈ Σ-*i* (*V*, *C*, *ε*) (*n* − *1*)
using *M-i.simps* **by** *blast*
then have *justification m* ∈ Σ-*i* (*V*, *C*, *ε*) *0*
apply (*induction n*)
apply *simp*
by (*smt M-i.simps One-nat-def Params.Σi-subset-Mi Pow-iff Suc-pred* ‹*m* ∈ *justification m*› *add.right-neutral add-Suc-right diff-Suc-1 mem-Collect-eq not-gr0 subsetCE subsetCE*)
then have *justification m* ∈ {∅}
by *simp*
then show *False*
using ‹*m* ∈ *justification m*› **by** *blast*
qed

lemma (**in** *Protocol*) *message-cannot-justify-itself* :
(∀ *m* ∈ *M*. ¬ *justified m m*)
proof −
have *irrefl message-justification*
using *irreflexivity-of-justifications* **by** *simp*
then show *?thesis*
by (*simp add*: *irreflexivity-of-justifications irrefl-def message-justification-def*)
qed

lemma (**in** *Protocol*) *justification-is-strict-partial-order-on-M* :
strict-partial-order message-justification
apply (*simp add*: *strict-partial-order-def*)
by (*simp add*: *irreflexivity-of-justifications transitivity-of-justifications*)

lemma (**in** *Protocol*) *monotonicity-of-justifications* :
$\forall\ m\ m'\ \sigma.\ m \in M \wedge \sigma \in \Sigma \wedge$ *justified* $m'\ m \longrightarrow$ *justification* $m' \subseteq$ *justification* m
apply *simp*
by (*meson M-type justified-def message-in-state-is-valid state-is-in-pow-M-i*)

lemma (**in** *Protocol*) *strict-monotonicity-of-justifications* :
$\forall\ m\ m'\ \sigma.\ m \in M \wedge \sigma \in \Sigma \wedge$ *justified* $m'\ m \longrightarrow$ *justification* $m' \subset$ *justification* m
by (*metis M-type message-cannot-justify-itself justified-def message-in-state-is-valid monotonicity-of-justifications psubsetI*)

lemma (**in** *Protocol*) *justification-implies-different-messages* :
$\forall\ m\ m'.\ m \in M \wedge m' \in M \longrightarrow$ *justified* $m'\ m \longrightarrow m \neq m'$
using *message-cannot-justify-itself* **by** *auto*

lemma (**in** *Protocol*) *only-valid-message-is-justified* :
$\forall\ m \in M.\ \forall\ m'.$ *justified* $m'\ m \longrightarrow m' \in M$
apply (*simp add*: *justified-def*)
using *Params.M-type message-in-state-is-valid* **by** *blast*

lemma (**in** *Protocol*) *justified-message-exists-in-M-i-n-minus-1* :
$\forall\ n\ m\ m'.\ n \in \mathbb{N}$
$\longrightarrow$ *justified* $m'\ m$
$\longrightarrow m \in M\text{-}i\ (V,\ C,\ \varepsilon)\ n$
$\longrightarrow\ m' \in M\text{-}i\ (V,\ C,\ \varepsilon)\ (n - 1)$
proof −
have $\forall\ n\ m\ m'.$ *justified* $m'\ m$
$\longrightarrow m \in M\text{-}i\ (V,\ C,\ \varepsilon)\ n$
$\longrightarrow m \in M \wedge m' \in M$
$\longrightarrow m' \in M\text{-}i\ (V,\ C,\ \varepsilon)\ (n - 1)$
apply (*rule*, *rule*, *rule*, *rule*, *rule*, *rule*)
proof −
fix $n\ m\ m'$
assume *justified* $m'\ m$
assume $m \in M\text{-}i\ (V,\ C,\ \varepsilon)\ n$
assume $m \in M \wedge m' \in M$
then have *justification* $m \in \Sigma\text{-}i\ (V,C,\varepsilon)\ n$
using *M-i.simps* $\langle m \in M\text{-}i\ (V,\ C,\ \varepsilon)\ n\rangle$ **by** *blast*
then have *justification* $m \in$ *Pow* $(M\text{-}i\ (V,C,\varepsilon)\ (n - 1))$
by (*metis* (*no-types*, *lifting*) *Suc-diff-Suc* Σ*-i.simps*(*1*) Σ*i-subset-Mi* ⟨*justified* $m'\ m$⟩ *add-leE diff-add diff-le-self empty-iff justified-def neq0-conv plus-1-eq-Suc singletonD subsetCE*)
show $m' \in M\text{-}i\ (V,\ C,\ \varepsilon)\ (n - 1)$
using ⟨*justification* $m \in$ *Pow* $(M\text{-}i\ (V,\ C,\ \varepsilon)\ (n - 1))$⟩ ⟨*justified* $m'\ m$⟩ *justified-def* **by** *auto*
qed
then show *?thesis*
by (*metis* (*no-types*, *lifting*) *M-def UN-I only-valid-message-is-justified*)

qed

lemma (**in** *Protocol*) *monotonicity-of-card-of-justification* :
$\forall\ m\ m'.\ m \in M$
$\longrightarrow$ *justified* m' m
$\longrightarrow$ *card* (*justification* m') $<$ *card* (*justification* m)
by (*meson M-type Protocol.strict-monotonicity-of-justifications Protocol-axioms justification-is-finite psubset-card-mono*)

lemma (**in** *Protocol*) *justification-is-well-founded-on-M* :
wfp-on justified M
proof (*rule ccontr*)
assume $\neg$ *wfp-on justified M*
then have $\exists f.\ \forall i.\ f\ i \in M \wedge$ *justified* (f (*Suc* i)) ($f\ i$)
by (*simp add*: *wfp-on-def*)
then obtain f **where** $\forall i.\ f\ i \in M \wedge$ *justified* (f (*Suc* i)) ($f\ i$) **by** *auto*
have $\forall\ i.$ *card* (*justification* ($f\ i$)) $\leq$ *card* (*justification* (f *0*)) $- i$
apply (*rule*)
proof $-$
fix i
have *card* (*justification* (f (*Suc* i))) $<$ *card* (*justification* ($f\ i$))
using ‹$\forall i.\ f\ i \in M \wedge$ *justified* (f (*Suc* i)) ($f\ i$)› **by** (*simp add*: *monotonicity-of-card-of-justification*)
show *card* (*justification* ($f\ i$)) $\leq$ *card* (*justification* (f *0*)) $- i$
apply (*induction* i)
apply *simp*
using ‹*card* (*justification* (f (*Suc* i))) $<$ *card* (*justification* ($f\ i$))›
by (*smt Suc-diff-le* ‹$\forall i.\ f\ i \in M \wedge$ *justified* (f (*Suc* i)) ($f\ i$)› *diff-Suc-Suc diff-is-0-eq le-iff-add less-Suc-eq-le less-imp-le monotonicity-of-card-of-justification not-less-eq-eq trans-less-add1*)
qed
then have $\exists\ i.\ i =$ *card* (*justification* (f *0*)) $+$ *Suc 0* $\wedge$ *card* (*justification* ($f\ i$)) $\leq$ *card* (*justification* (f *0*)) $- i$
by *blast*
then show *False*
using *le-0-eq le-simps*(*2*) *linorder-not-le monotonicity-of-card-of-justification nat-diff-split order-less-imp-le*
by (*metis* ‹$\forall i.\ f\ i \in M \wedge$ *justified* (f (*Suc* i)) ($f\ i$)› *add.right-neutral add-Suc-right*)
qed

lemma (**in** *Protocol*) *subset-of-M-have-minimal-of-justification* :
$\forall\ S \subseteq M.\ S \neq \emptyset \longrightarrow (\exists$ *m-min* $\in S.\ \forall\ m.$ *justified m m-min* $\longrightarrow m \notin S)$
by (*metis justification-is-well-founded-on-M wfp-on-imp-has-min-elt wfp-on-mono*)

end

2 Safety Proof

theory *ConsensusSafety*

imports *Main CBCCasper Libraries/LaTeXsugar*

begin

fun (**in** *Protocol*) *futures* :: *state* ⇒ *state set*
  where
    $futures\ \sigma = \{\sigma' \in \Sigma t.\ is\text{-}future\text{-}state\ (\sigma, \sigma')\}$

lemma (**in** *Protocol*) *monotonic-futures* :
  $\forall\ \sigma'\ \sigma.\ \sigma' \in \Sigma t \wedge \sigma \in \Sigma t$
   $\longrightarrow \sigma' \in futures\ \sigma \longleftrightarrow futures\ \sigma' \subseteq futures\ \sigma$
  by *auto*

theorem (**in** *Protocol*) *two-party-common-futures* :
  $\forall\ \sigma 1\ \sigma 2.\ \sigma 1 \in \Sigma t \wedge \sigma 2 \in \Sigma t$
  $\longrightarrow (\sigma 1 \cup \sigma 2) \in \Sigma t$
  $\longrightarrow futures\ \sigma 1 \cap futures\ \sigma 2 \neq \emptyset$
  by *auto*

theorem (**in** *Protocol*) *n-party-common-futures* :
  $\forall\ \sigma\text{-}set.\ \sigma\text{-}set \subseteq \Sigma t$
  $\longrightarrow \bigcup\ \sigma\text{-}set \in \Sigma t$
  $\longrightarrow \bigcap\ \{futures\ \sigma \mid \sigma.\ \sigma \in \sigma\text{-}set\} \neq \emptyset$
  by *auto*

fun (**in** *Protocol*) *state-property-is-decided* :: (*state-property* ∗ *state*) ⇒ *bool*
  where
    $state\text{-}property\text{-}is\text{-}decided\ (p, \sigma) = (\forall\ \sigma' \in futures\ \sigma\ .\ p\ \sigma')$

lemma (**in** *Protocol*) *forward-consistency* :
  $\forall\ \sigma'\ \sigma.\ \sigma' \in \Sigma t \wedge \sigma \in \Sigma t$
  $\longrightarrow \sigma' \in futures\ \sigma$
  $\longrightarrow state\text{-}property\text{-}is\text{-}decided\ (p, \sigma)$
  $\longrightarrow state\text{-}property\text{-}is\text{-}decided\ (p, \sigma')$
  apply *simp*
  by *auto*

fun *state-property-not* :: *state-property* $\Rightarrow$ *state-property*
  where
    state-property-not $p = (\lambda\sigma.\ (\neg\ p\ \sigma))$

lemma (**in** *Protocol*) *backword-consistency* :
  $\forall\ \sigma'\ \sigma.\ \sigma' \in \Sigma t \wedge \sigma \in \Sigma t$
  $\longrightarrow \sigma' \in$ *futures* σ
  $\longrightarrow$ *state-property-is-decided* (p, σ')
  $\longrightarrow \neg$*state-property-is-decided* (*state-property-not* p, σ)
  apply *simp*
  by *auto*

theorem (**in** *Protocol*) *two-party-consensus-safety* :
  $\forall\ \sigma 1\ \sigma 2.\ \sigma 1 \in \Sigma t \wedge \sigma 2 \in \Sigma t$
  $\longrightarrow (\sigma 1 \cup \sigma 2) \in \Sigma t$
  $\longrightarrow \neg($*state-property-is-decided* $(p, \sigma 1) \wedge$ *state-property-is-decided* (*state-property-not* p, $\sigma 2$))
  by *auto*

fun (**in** *Protocol*) *state-properties-are-inconsistent* :: *state-property set* $\Rightarrow$ *bool*
  where
    state-properties-are-inconsistent p-set $= (\forall\ \sigma \in \Sigma.\ \neg\ (\forall\ p \in$ *p-set*$.\ p\ \sigma))$

fun (**in** *Protocol*) *state-properties-are-consistent* :: *state-property set* $\Rightarrow$ *bool*
  where
    state-properties-are-consistent p-set $= (\exists\ \sigma \in \Sigma.\ \forall\ p \in$ *p-set*$.\ p\ \sigma)$

fun (**in** *Protocol*) *state-property-decisions* :: *state* $\Rightarrow$ *state-property set*
  where
    state-property-decisions $\sigma = \{p.$ *state-property-is-decided* $(p, \sigma)\}$

theorem (**in** *Protocol*) *n-party-safety-for-state-properties* :
  $\forall\ \sigma$*-set*. σ*-set* $\subseteq \Sigma t$
  $\longrightarrow \bigcup\ \sigma$*-set* $\in \Sigma t$
  $\longrightarrow$ *state-properties-are-consistent* $(\bigcup\ \{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\})$
  apply *rule+*
proof−
  fix σ*-set*
  assume σ*-set*: σ*-set* $\subseteq \Sigma t$

  assume $\bigcup\ \sigma$*-set* $\in \Sigma t$
  hence $\bigcap\ \{$*futures* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\} \neq \emptyset$

using *σ-set* **by** *auto*
hence $\exists\sigma\in\Sigma t.\ \sigma \in \bigcap\ \{$*futures* $\sigma \mid \sigma.\ \sigma \in$ *σ-set*$\}$
using ‹$\bigcup$*σ-set* $\in \Sigma t$› **by** *fastforce*
hence $\exists\sigma\in\Sigma t.\ \forall s\in$*σ-set*. $\sigma \in$ *futures s*
by *blast*
hence $\exists\sigma\in\Sigma t.$ ($\forall s\in$*σ-set*. $\sigma \in$ *futures s*) $\wedge$ ($\forall s\in$*σ-set*. $\sigma \in$ *futures s* $\longrightarrow$ ($\forall p.$ *state-property-is-decided* $(p,s) \longrightarrow$ *state-property-is-decided* (p,σ)))
by (*simp add*: *subset-eq*)
hence $\exists\sigma\in\Sigma t.\ \forall s\in$*σ-set*. ($\forall p.$ *state-property-is-decided* $(p,s) \longrightarrow$ *state-property-is-decided* (p,σ))
by *blast*
hence $\exists\sigma\in\Sigma t.\ \forall s\in$*σ-set*. ($\forall p \in$ *state-property-decisions s*. *state-property-is-decided* (p,σ))
by *simp*
hence $\exists\sigma\in\Sigma t.\ \forall p\in\bigcup\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in$ *σ-set*$\}$. *state-property-is-decided* (p,σ)
proof−
obtain σ **where** $\sigma \in \Sigma t\ \forall s\in$*σ-set*. ($\forall p \in$ *state-property-decisions s*. *state-property-is-decided* (p,σ))
using ‹$\exists\sigma\in\Sigma t.\ \forall s\in$*σ-set*. $\forall p\in$*state-property-decisions s*. *state-property-is-decided* (p, σ)› **by** *blast*
have $\forall p\in\bigcup\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in$ *σ-set*$\}$. *state-property-is-decided* (p,σ)
using ‹$\forall s\in$*σ-set*. $\forall p\in$*state-property-decisions s*. *state-property-is-decided* (p, σ)› **by** *fastforce*
thus *?thesis*
using ‹$\sigma \in \Sigma t$› **by** *blast*
qed
hence $\exists\sigma\in\Sigma t.\ \forall p\in\bigcup\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in$ *σ-set*$\}$. $\forall\sigma'\in$*futures* $\sigma.\ p\ \sigma'$
by *simp*
show *state-properties-are-consistent* ($\bigcup\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in$ *σ-set*$\}$)
by (*metis* (*mono-tags*, *lifting*) *Σt-def* ‹$\exists\sigma\in\Sigma t.\ \forall p\in\bigcup\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in$ *σ-set*$\}$. $\forall\sigma'\in$*futures* $\sigma.\ p\ \sigma'$› *mem-Collect-eq monotonic-futures order-refl state-properties-are-consistent.simps*)
qed

fun (**in** *Protocol*) *naturally-corresponding-state-property* :: *consensus-value-property* $\Rightarrow$ *state-property*
where
naturally-corresponding-state-property q = ($\lambda\sigma.\ \forall\ c \in \varepsilon\ \sigma.\ q\ c$)

fun (**in** *Protocol*) *consensus-value-properties-are-consistent* :: *consensus-value-property set* $\Rightarrow$ *bool*
where

consensus-value-properties-are-consistent q-set $= (\exists\ c \in C.\ \forall\ q \in$ *q-set*. $q\ c)$

lemma (**in** *Protocol*) *naturally-corresponding-consistency* :
$\forall$ *q-set*. *state-properties-are-consistent* {*naturally-corresponding-state-property q* | *q*. $q \in$ *q-set*}
$\longrightarrow$ *consensus-value-properties-are-consistent q-set*
apply (*rule*, *rule*)
proof −
fix *q-set*

have
state-properties-are-consistent {*naturally-corresponding-state-property q* | *q*. $q \in$ *q-set*}
$\longrightarrow (\exists\ \sigma \in \Sigma.\ \forall\ p \in \{\lambda\sigma'.\ \forall\ c \in \varepsilon\ \sigma'.\ q\ c \mid q.\ q \in$ *q-set*$\}.\ p\ \sigma)$
by *simp*
moreover have
$(\exists\ \sigma \in \Sigma.\ \forall\ p \in \{\lambda\sigma'.\ \forall\ c \in \varepsilon\ \sigma'.\ q\ c \mid q.\ q \in$ *q-set*$\}.\ p\ \sigma)$
$\longrightarrow (\exists\ \sigma \in \Sigma.\ \forall\ q' \in$ *q-set*. $(\lambda\sigma'.\ \forall\ c \in \varepsilon\ \sigma'.\ q'\ c)\ \sigma)$
by (*metis* (*mono-tags*, *lifting*) *mem-Collect-eq*)
moreover have
$(\exists\ \sigma \in \Sigma.\ \forall\ q \in$ *q-set*. $(\lambda\sigma'.\ \forall\ c \in \varepsilon\ \sigma'.\ q\ c)\ \sigma)$
$\longrightarrow (\exists\ \sigma \in \Sigma.\ \forall\ q' \in$ *q-set*. $\forall\ c \in \varepsilon\ \sigma.\ q'\ c)$
by *blast*
moreover have
$(\exists\ \sigma \in \Sigma.\ \forall\ q \in$ *q-set*. $\forall\ c \in \varepsilon\ \sigma.\ q\ c)$
$\longrightarrow (\exists\ \sigma \in \Sigma.\ \forall\ c \in \varepsilon\ \sigma.\ \forall\ q' \in$ *q-set*. $q'\ c)$
by *blast*
moreover have
$(\exists\ \sigma \in \Sigma.\ \forall\ c \in \varepsilon\ \sigma.\ \forall\ q \in$ *q-set*. $q\ c)$
$\longrightarrow (\exists\ \sigma \in \Sigma.\ \exists\ c \in \varepsilon\ \sigma.\ \forall\ q' \in$ *q-set*. $q'\ c)$
by (*meson all-not-in-conv estimates-are-non-empty*)
moreover have
$(\exists\ \sigma \in \Sigma.\ \exists\ c \in \varepsilon\ \sigma.\ \forall\ q \in$ *q-set*. $q\ c)$
$\longrightarrow (\exists\ c \in C.\ \forall\ q' \in$ *q-set*. $q'\ c)$
using *is-valid-estimator-def* ε*-type* **by** *fastforce*
ultimately show
state-properties-are-consistent {*naturally-corresponding-state-property q* |*q*. $q \in$ *q-set*}
$\Longrightarrow$ *consensus-value-properties-are-consistent q-set*
by *simp*
qed

fun (**in** *Protocol*) *consensus-value-property-is-decided* :: (*consensus-value-property* $*$ *state*) $\Rightarrow$ *bool*
where
consensus-value-property-is-decided (q, σ)
= *state-property-is-decided* (*naturally-corresponding-state-property q*, σ)

fun (**in** *Protocol*) *consensus-value-property-decisions* :: *state* $\Rightarrow$ *consensus-value-property set*
  where
    consensus-value-property-decisions $\sigma = \{q.$ *consensus-value-property-is-decided* $(q, \sigma)\}$

theorem (**in** *Protocol*) *n-party-safety-for-consensus-value-properties* :
$\forall$ σ*-set*. σ*-set* $\subseteq \Sigma t$
$\longrightarrow \bigcup$ σ*-set* $\in \Sigma t$
$\longrightarrow$ *consensus-value-properties-are-consistent* $(\bigcup$ $\{$*consensus-value-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\})$
  apply (*rule*, *rule*, *rule*)
proof –
  fix σ*-set*
  assume σ*-set* $\subseteq \Sigma t$

  assume $\bigcup$ σ*-set* $\in \Sigma t$
  hence *state-properties-are-consistent* $(\bigcup$ $\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\})$
    using ‹σ*-set* $\subseteq \Sigma t$› *n-party-safety-for-state-properties* **by** *auto*
  hence *state-properties-are-consistent* $\{p \in \bigcup$ $\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\}.\ \exists$ $q.\ p =$ *naturally-corresponding-state-property* $q\}$
    apply *simp*
    by *meson*
  hence *state-properties-are-consistent* $\{$*naturally-corresponding-state-property* $q \mid q.$ *naturally-corresponding-state-property* $q \in \bigcup$ $\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\}\}$
    by (*smt Collect-cong*)
  hence *consensus-value-properties-are-consistent* $\{q.$ *naturally-corresponding-state-property* $q \in \bigcup$ $\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\}\}$
    using *naturally-corresponding-consistency*
  proof –
    show *?thesis*
     by (*metis* (*no-types*) *Setcompr-eq-image* ‹$\forall$ *q-set*. *state-properties-are-consistent* $\{$*naturally-corresponding-state-property* $q \mid q.\ q \in$ *q-set*$\} \longrightarrow$ *consensus-value-properties-are-consistent q-set*› ‹*state-properties-are-consistent* $\{$*naturally-corresponding-state-property* $q \mid q.$ *naturally-corresponding-state-property* $q \in \bigcup\{$*state-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\}\}$› *setcompr-eq-image*)
  qed
  hence *consensus-value-properties-are-consistent* $(\bigcup$ $\{$*consensus-value-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\})$
    apply *simp*
    by (*smt mem-Collect-eq*)
  thus
   consensus-value-properties-are-consistent $(\bigcup$ $\{$*consensus-value-property-decisions* $\sigma \mid \sigma.\ \sigma \in \sigma$*-set*$\})$

by *simp*
qed

end

3 Latest Message

theory *LatestMessage*

imports *Main CBCCasper Libraries/LaTeXsugar*

begin

definition *later* :: $(message * state) \Rightarrow message\ set$
where
$later = (\lambda(m, \sigma).\ \{m' \in \sigma.\ justified\ m\ m'\})$

lemma (**in** *Protocol*) *later-type* :
$\forall\ \sigma\ m.\ \sigma \in \Sigma \wedge m \in M \longrightarrow later\ (m, \sigma) \subseteq M$
apply (*simp add*: *later-def*)
using *state-is-subset-of-M* **by** *auto*

definition *from-sender* :: $(validator * state) \Rightarrow message\ set$
where
$from\text{-}sender = (\lambda(v, \sigma).\ \{m \in \sigma.\ sender\ m = v\})$

lemma (**in** *Protocol*) *from-sender-type* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \longrightarrow from\text{-}sender\ (v, \sigma) \subseteq M$
apply (*simp add*: *from-sender-def*)
using *state-is-subset-of-M* **by** *auto*

lemma (**in** *Protocol*) *messages-from-observed-validator-is-non-empty* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in observed\ \sigma \longrightarrow from\text{-}sender\ (v, \sigma) \neq \emptyset$
apply (*simp add*: *observed-def from-sender-def*)
by *auto*

lemma (**in** *Protocol*) *messages-from-validator-is-finite* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V\sigma \longrightarrow finite\ (from\text{-}sender\ (v, \sigma))$
by (*simp add*: *from-sender-def state-is-finite*)

definition *from-group* :: (*validator set* $*$ *state*) $\Rightarrow$ *state*
where
from-group = $(\lambda(v\text{-}set,\ \sigma).\ \{m \in \sigma.\ sender\ m \in v\text{-}set\})$

lemma (**in** *Protocol*) *from-group-type* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v\text{-}set \subseteq V \longrightarrow from\text{-}group\ (v\text{-}set,\ \sigma) \subseteq M$
apply (*simp add*: *from-group-def*)
using *state-is-subset-of-M* **by** *auto*

definition *later-from* :: (*message* $*$ *validator* $*$ *state*) $\Rightarrow$ *message set*
where
later-from = $(\lambda(m,\ v,\ \sigma).\ later\ (m,\ \sigma) \cap from\text{-}sender\ (v,\ \sigma))$

lemma (**in** *Protocol*) *later-from-type* :
$\forall\ \sigma\ v\ m.\ \sigma \in \Sigma \wedge v \in V \wedge m \in M \longrightarrow later\text{-}from\ (m,\ v,\ \sigma) \subseteq M$
apply (*simp add*: *later-from-def*)
using *later-type from-sender-type* **by** *auto*

definition *latest-messages* :: *state* $\Rightarrow$ (*validator* $\Rightarrow$ *state*)
where
$latest\text{-}messages\ \sigma\ v = \{m \in from\text{-}sender\ (v,\ \sigma).\ later\text{-}from\ (m,\ v,\ \sigma) = \emptyset\}$

lemma (**in** *Protocol*) *latest-messages-type* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \longrightarrow latest\text{-}messages\ \sigma\ v \subseteq M$
apply (*simp add*: *latest-messages-def later-from-def*)
using *from-sender-type* **by** *auto*

lemma (**in** *Protocol*) *latest-messages-from-non-observed-validator-is-empty* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \wedge v \notin observed\ \sigma \longrightarrow latest\text{-}messages\ \sigma\ v = \emptyset$
by (*simp add*: *latest-messages-def observed-def later-def from-sender-def*)

definition *observed-non-equivocating-validators* :: *state* $\Rightarrow$ *validator set*
where
$observed\text{-}non\text{-}equivocating\text{-}validators\ \sigma = observed\ \sigma - equivocating\text{-}validators\ \sigma$

lemma (**in** *Protocol*) *observed-non-equivocating-validators-type* :
$\forall\ \sigma \in \Sigma.\ observed\text{-}non\text{-}equivocating\text{-}validators\ \sigma \subseteq V$
apply (*simp add*: *observed-non-equivocating-validators-def*)
using *observed-type equivocating-validators-type* **by** *auto*

lemma (**in** *Protocol*) *justification-is-well-founded-on-messages-from-validator*:
$\forall\ \sigma \in \Sigma.\ (\forall\ v \in V.\ \ wfp\text{-}on\ justified\ (from\text{-}sender\ (v,\ \sigma)))$
using *justification-is-well-founded-on-M from-sender-type wfp-on-subset* **by** *blast*

lemma (**in** *Protocol*) *justification-is-total-on-messages-from-non-equivocating-validator*:
$\forall\ \sigma \in \Sigma.\ (\forall\ v \in V.\ v \notin$ *equivocating-validators* $\sigma \longrightarrow$ *Relation.total-on* (*from-sender* (v, σ)) *message-justification*)
proof −
have $\forall\ m1\ m2\ \sigma\ v.\ v \in V \wedge \sigma \in \Sigma \wedge \{m1, m2\} \subseteq$ *from-sender* $(v, \sigma) \longrightarrow$ *sender* $m1 =$ *sender* $m2$
by (*simp add*: *from-sender-def*)
then have $\forall\ \sigma \in \Sigma.\ (\forall\ v \in V.\ v \notin$ *equivocating-validators* σ
$\longrightarrow (\forall\ m1\ m2.\ \{m1, m2\} \subseteq$ *from-sender* $(v, \sigma) \longrightarrow m1 = m2 \vee$ *justified* $m1\ m2 \vee$ *justified* $m2\ m1$))
apply (*simp add*: *equivocating-validators-def is-equivocating-def equivocation-def from-sender-def observed-def*)
by *blast*
then show *?thesis*
apply (*simp add*: *Relation.total-on-def message-justification-def*)
using *from-sender-type* **by** *blast*
qed

lemma (**in** *Protocol*) *justification-is-strict-linear-order-on-messages-from-non-equivocating-validator*:
$\forall\ \sigma \in \Sigma.\ (\forall\ v \in V.\ v \notin$ *equivocating-validators* $\sigma \longrightarrow$ *strict-linear-order-on* (*from-sender* (v, σ)) *message-justification*)
by (*simp add*: *strict-linear-order-on-def justification-is-total-on-messages-from-non-equivocating-validator*

irreflexivity-of-justifications transitivity-of-justifications)

lemma (**in** *Protocol*) *justification-is-strict-well-order-on-messages-from-non-equivocating-validator*:
$\forall\ \sigma \in \Sigma.\ (\forall\ v \in V.\ v \notin$ *equivocating-validators* σ
$\longrightarrow$ *strict-linear-order-on* (*from-sender* (v, σ)) *message-justification* $\wedge$ *wfp-on justified* (*from-sender* (v, σ)))
using *justification-is-well-founded-on-messages-from-validator*
justification-is-strict-linear-order-on-messages-from-non-equivocating-validator

by *blast*

lemma (**in** *Protocol*) *latest-message-is-maximal-element-of-justification* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \longrightarrow$ *latest-messages* $\sigma\ v = \{m.$ *maximal-on* (*from-sender* (v, σ)) *message-justification* $m\}$
apply (*simp add*: *latest-messages-def later-from-def later-def message-justification-def maximal-on-def*)
using *from-sender-type* **apply** *auto*
apply (*metis* (*no-types*, *lifting*) *IntI empty-iff from-sender-def mem-Collect-eq prod.simps*(*2*))
by *blast*

lemma (**in** *Protocol*) *observed-non-equivocating-validators-have-one-latest-message*:
$\forall\ \sigma \in \Sigma.\ (\forall\ v \in$ *observed-non-equivocating-validators* σ. *is-singleton* (*latest-messages* $\sigma\ v$))

apply (*simp add*: *observed-non-equivocating-validators-def*)
proof −
have $\forall\ \sigma \in \Sigma.\ (\forall\ v \in$ *observed* $\sigma -$ *equivocating-validators* σ. *is-singleton* $\{m.$ *maximal-on* (*from-sender* (v, σ)) *message-justification* $m\})$
using
messages-from-observed-validator-is-non-empty
messages-from-validator-is-finite
observed-type
equivocating-validators-def
justification-is-strict-linear-order-on-messages-from-non-equivocating-validator
strict-linear-order-on-finite-non-empty-set-has-one-maximum
maximal-and-maximum-coincide-for-strict-linear-order
by (*smt Collect-cong DiffD1 DiffD2 set-mp*)
then show $\forall \sigma{\in}\Sigma.\ \forall v{\in}$*observed* $\sigma -$ *equivocating-validators* σ. *is-singleton* (*latest-messages* σ v)
using *latest-message-is-maximal-element-of-justification*
observed-non-equivocating-validators-def observed-non-equivocating-validators-type

by *fastforce*
qed

definition *latest-estimates* :: *state* ⇒ *validator* ⇒ *consensus-value set*
where
latest-estimates σ v = $\{$*est* m $\mid$ m. $m \in$ *latest-messages* σ $v\}$

lemma (**in** *Protocol*) *latest-estimates-type* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \longrightarrow$ *latest-estimates* σ $v \subseteq C$
using *M-type Protocol.latest-messages-type Protocol-axioms latest-estimates-def*
by *fastforce*

lemma (**in** *Protocol*) *latest-estimates-from-non-observed-validator-is-empty* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \wedge v \notin$ *observed* $\sigma \longrightarrow$ *latest-estimates* σ $v = \emptyset$
using *latest-estimates-def latest-messages-from-non-observed-validator-is-empty*
by *auto*

definition *latest-messages-from-non-equivocating-validators* :: *state* ⇒ *validator* ⇒ *message set*
where
latest-messages-from-non-equivocating-validators σ v = (*if is-equivocating* σ v *then* $\emptyset$ *else latest-messages* σ v)

lemma (**in** *Protocol*) *latest-messages-from-non-equivocating-validators-type* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \longrightarrow$ *latest-messages-from-non-equivocating-validators* $\sigma\ v \subseteq M$
by (*simp add*: *latest-messages-type latest-messages-from-non-equivocating-validators-def*)

definition *latest-estimates-from-non-equivocating-validators* :: *state* $\Rightarrow$ *validator* $\Rightarrow$ *consensus-value set*
where
latest-estimates-from-non-equivocating-validators $\sigma\ v = \{$*est* $m \mid m.\ m \in$ *latest-messages-from-non-equivocating-validators* $\sigma\ v\}$

lemma (**in** *Protocol*) *latest-estimates-from-non-equivocating-validators-type* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \longrightarrow$ *latest-estimates-from-non-equivocating-validators* $\sigma\ v \subseteq C$
using *Protocol.latest-estimates-type Protocol-axioms latest-estimates-def latest-estimates-from-non-equivocati*
latest-messages-from-non-equivocating-validators-def **by** *auto*

lemma (**in** *Protocol*) *latest-estimates-from-non-equivocating-validators-from-non-observed-validator-is-empty* :
$\forall\ \sigma\ v.\ \sigma \in \Sigma \wedge v \in V \wedge v \notin$ *observed* $\sigma \longrightarrow$ *latest-estimates-from-non-equivocating-validators* $\sigma\ v = \emptyset$
by (*simp add*: *latest-estimates-from-non-equivocating-validators-def latest-messages-from-non-equivocating-va*
latest-messages-from-non-observed-validator-is-empty)

end