ivy-method-in-isabelle

nano

April 1, 2024

Contents

theory *PBFT*
  imports *Main HOL−Eisbach.Eisbach*
begin

method *rewrite-funs* = (*simp* (*no-asm-use*) *only*: *fun-upd-apply*[*abs-def*] *fun-eq-iff if-split-asm*)
  — rewrites function-update and function equality terms to first-order equality.

1 First-Order Abstraction

locale *byz-quorums* =
  fixes *byz-member* :: *'n* ⇒ *'q1* ⇒ *bool* (**infix** $\in_b$ *50*)
    and *is-good* :: *'n* ⇒ *bool*
    and *good-member* :: *'n* ⇒ *'q2* ⇒ *bool* (**infix** $\in_g$ *50*)
  assumes
    ⋀ *q n* . *n* $\in_g$ *q* ⟹ *is-good n*
      — good quorums contain only good nodes
    and ⋀ *q1 q2* . ∃ *n* . *n* $\in_g$ *q1* ∧ *n* $\in_g$ *q2*
      — good quorums intersect
    and ⋀ *q1* . ∃ *q2* . ∀ *n* . *n* $\in_g$ *q2* ⟶ *n* $\in_b$ *q1*
      — every byz quorum contains at least one good quorum

locale *fol-bft* = *byz-quorums* + *linorder less-eq less* **for**
  less-eq :: *'r* ⇒ *'r* ⇒ *bool* (**infix** ≤ *50*) **and** *less* (**infix** < *50*) +
fixes *bot* :: *'r*
  — the special bottom round
begin

no-notation
ord-class.less-eq (*op* $\leq$) **and**
ord-class.less-eq ((-/ $\leq$ -) [*51*, *51*] *50*) **and**
ord-class.less (*op* $<$) **and**
ord-class.less ((-/ $<$ -) [*51*, *51*] *50*) **and**
Set.member (*op* $\in$) **and**
Set.member ((-/ $\in$ -) [*51*, *51*] *50*)

lemma *False*
— checking for inconsistent hypothesis
nitpick[*expect*=*genuine*, *verbose*=*true*] **oops**

lemmas *fol-bft-assms* = *fol-bft-axioms fol-bft-def class.linorder-def class.order-def*
class.preorder-def class.order-axioms-def class.linorder-axioms-def byz-quorums-def

end

2 Abstract PBFT Model

We use propose messages to verify 1b messages as in Lamport's version of PBFT. We do not model pre-prepare messages.

context *fol-bft*
begin

definition *byz-send* **where**
— byzantine nodes can do anything but impersonate non-byz nodes
byz-send vote-msg proposal start-round-msg left-round joined-round
vote-msg′ proposal′ start-round-msg′ left-round′ joined-round′ $\equiv$
$\forall$ *n* . *is-good n* $\longrightarrow$ (
vote-msg′ n = *vote-msg n*
$\wedge$ *left-round′ n* = *left-round n*
$\wedge$ *joined-round′ n* = *joined-round n*
$\wedge$ *proposal′ n* = *proposal n*)

definition *join-round* **where**
join-round vote-msg proposal start-round-msg left-round joined-round
vote-msg′ proposal′ start-round-msg′ left-round′ joined-round′ n r $\equiv$
r $\neq$ *bot*
$\wedge$ *start-round-msg r*
$\wedge$ $\neg$ *left-round n r*
$\wedge$ ($\forall$ *n′ r′* . *left-round′ n′ r′* = ((*left-round n′ r′*) $\vee$ (*n′* = *n* $\wedge$ *r′* $<$ *r*)))
$\wedge$ *joined-round′* = *joined-round*(*n* := (*joined-round n*)(*r* := *True*))
$\wedge$ *vote-msg′* = *vote-msg* $\wedge$ *proposal′* = *proposal* $\wedge$ *start-round-msg′* = *start-round-msg*

definition *propose* **where**
propose vote-msg proposal start-round-msg left-round joined-round
vote-msg′ proposal′ start-round-msg′ left-round′ joined-round′ n q r $\equiv$
($\forall$ *v* . $\neg$ *proposal n r v*)

$\wedge\ r \neq bot$
$\wedge\ (\forall\ n\ .\ n \in_b q \longrightarrow joined\text{-}round\ n\ r)$
$\wedge\ (\exists\ v\ .$
$(\ (\forall\ n\ r'\ v\ .\ n \in_b q \wedge r' < r \longrightarrow \neg vote\text{-}msg\ n\ r'\ v)$
$\vee\ (\exists\ rmax\ .\ rmax \neq bot \wedge rmax < r$
$\wedge\ (\forall\ n\ v'\ .\ n \in_b q \wedge vote\text{-}msg\ n\ rmax\ v' \longrightarrow v = v')$
$\wedge\ (\forall\ n\ r'\ v\ .\ n \in_b q \wedge vote\text{-}msg\ n\ r'\ v \wedge r' < r \longrightarrow r' \leq rmax)$
$\wedge\ (\exists\ n\ .\ is\text{-}good\ n \wedge proposal\ n\ rmax\ v))\ )$
$\wedge\ proposal' = proposal(n := (proposal\ n)(r := (proposal\ n\ r)(v := True)))\ )$
$\wedge\ vote\text{-}msg' = vote\text{-}msg \wedge start\text{-}round\text{-}msg' = start\text{-}round\text{-}msg \wedge left\text{-}round = left\text{-}round' \wedge joined\text{-}round' = joined\text{-}round$

definition *vote* **where**
vote vote-msg proposal start-round-msg left-round joined-round vote-msg' proposal' start-round-msg' left-round' joined-round' n q r v $\equiv$
$r \neq bot$
$\wedge\ \neg\ left\text{-}round\ n\ r$
$\wedge\ (\exists\ q\ .\ \forall\ n\ .\ n \in_b q \longrightarrow proposal\ n\ r\ v)$
$\wedge\ vote\text{-}msg' = vote\text{-}msg(n := (vote\text{-}msg\ n)(r := (vote\text{-}msg\ n\ r)(v := True)))$
$\wedge\ proposal' = proposal \wedge left\text{-}round = left\text{-}round' \wedge joined\text{-}round' = joined\text{-}round \wedge start\text{-}round\text{-}msg' = start\text{-}round\text{-}msg$

definition *trans* **where**
trans vote-msg proposal start-round-msg left-round joined-round vote-msg' proposal' start-round-msg' left-round' joined-round' n r v q $\equiv$
join-round vote-msg proposal start-round-msg left-round joined-round vote-msg' proposal' start-round-msg' left-round' joined-round' n r
$\vee$ *propose vote-msg proposal start-round-msg left-round joined-round vote-msg' proposal' start-round-msg' left-round' joined-round' n q r*
$\vee$ *vote vote-msg proposal start-round-msg left-round joined-round vote-msg' proposal' start-round-msg' left-round' joined-round' n q r v*
$\vee$ *byz-send vote-msg proposal start-round-msg left-round joined-round vote-msg' proposal' start-round-msg' left-round' joined-round'*

3 Auxiliary Invariants and Safety Property

definition *inv1* **where**
inv1 vote-msg proposal start-round-msg left-round joined-round $\equiv \forall\ n\ r\ v\ .$
$(\forall\ r'\ .\ is\text{-}good\ n \wedge joined\text{-}round\ n\ r \wedge r' < r \longrightarrow left\text{-}round\ n\ r')$
$\wedge\ (is\text{-}good\ n \wedge vote\text{-}msg\ n\ r\ v \longrightarrow (\exists\ q\ .\ \forall\ n\ .\ n \in_g q \longrightarrow proposal\ n\ r\ v))$
$\wedge\ (\forall\ v'\ .\ is\text{-}good\ n \wedge proposal\ n\ r\ v \wedge proposal\ n\ r\ v' \longrightarrow v = v')$

definition *choosable-inv* **where**
choosable-inv vote-msg proposal start-round-msg left-round joined-round $\equiv$
$\forall\ r1\ r2\ v1\ v2\ n\ .\ r1 < r2 \wedge is\text{-}good\ n \wedge proposal\ n\ r2\ v2 \wedge v1 \neq v2 \longrightarrow$
$(\forall\ q2\ .\ \exists\ n2\ .\ n2 \in_g q2 \wedge left\text{-}round\ n2\ r1 \wedge \neg\ vote\text{-}msg\ n2\ r1\ v1)$

definition *safety* **where**
safety vote-msg proposal start-round-msg left-round joined-round $\equiv$

$\forall\ r\ r'\ q\ q'\ v\ v'\ .\ (\forall\ n\ .\ n \in_b q \longrightarrow \textit{vote-msg}\ n\ r\ v) \wedge (\forall\ n\ .\ n \in_b q' \longrightarrow \textit{vote-msg}\ n\ r'\ v')$
$\longrightarrow v = v'$

4 Safety proof

lemma *inv1*:
assumes *inv1 vote-msg proposal start-round-msg left-round joined-round* **and**
trans vote-msg proposal start-round-msg left-round joined-round
vote-msg' proposal' start-round-msg' left-round' joined-round' n r v q
shows *inv1 vote-msg' proposal' start-round-msg' left-round' joined-round'*
using *assms fol-bft-axioms*
unfolding *trans-def inv1-def byz-send-def join-round-def propose-def vote-def fol-bft-assms*
apply *rewrite-funs* **apply** *smt*
done

lemma *choosable*:
assumes *inv1 vote-msg proposal start-round-msg left-round joined-round*
choosable-inv vote-msg proposal start-round-msg left-round joined-round **and**
trans vote-msg proposal start-round-msg left-round joined-round
vote-msg' proposal' start-round-msg' left-round' joined-round' n r v q
shows *choosable-inv vote-msg' proposal' start-round-msg' left-round' joined-round'*
using *assms fol-bft-axioms*
unfolding *trans-def inv1-def byz-send-def join-round-def propose-def vote-def fol-bft-assms choosable-inv-def*
apply *rewrite-funs* **apply** *smt*
done

lemma *safety*:
assumes *inv1 vote-msg proposal start-round-msg left-round joined-round* **and**
choosable-inv vote-msg proposal start-round-msg left-round joined-round
shows *safety vote-msg proposal start-round-msg left-round joined-round*
using *assms fol-bft-axioms*
unfolding *safety-def inv1-def choosable-inv-def choosable-inv-def fol-bft-assms*
apply *auto* **apply** *metis*
done

end

end