MODULE *Config*

EXTENDS
Northbound,
Proposal,
Configuration,
Mastership,
Southbound

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

LOCAL INSTANCE *TLC*

$vars \triangleq \langle proposal, configuration, mastership, target \rangle$

Formal specification, constraints, and theorems.

$Init \triangleq$
$\land\ InitNorthbound$
$\land\ InitProposal$
$\land\ InitConfiguration$
$\land\ InitMastership$
$\land\ InitSouthbound$

$Next \triangleq$
$\lor\ \land\ NextNorthbound$
$\quad\land$ UNCHANGED $\langle\rangle$
$\lor\ \land\ NextProposal$
$\quad\land$ UNCHANGED $\langle\rangle$
$\lor\ \land\ NextConfiguration$
$\quad\land$ UNCHANGED $\langle proposal \rangle$
$\lor\ \land\ NextMastership$
$\quad\land$ UNCHANGED $\langle proposal, configuration \rangle$
$\lor\ \land\ NextSouthbound$
$\quad\land$ UNCHANGED $\langle proposal, configuration, mastership \rangle$

$Spec \triangleq Init \land \Box[Next]_{vars} \land \mathrm{WF}_{vars}(Next)$

$Order \triangleq$
$\forall\, i \in$ DOMAIN $proposal :$
$\land\ proposal[i].phase = ProposalCommit \land proposal[i].state \in \{ProposalComplete, ProposalFailed\} \Rightarrow$
$\forall\, j \in$ DOMAIN $proposal :$
$j < i \Rightarrow$

```
                ∧ proposal[j].phase = ProposalCommit ⇒ proposal[j].state ≠ ProposalInProgress
        ∧ proposal[i].phase = ProposalApply ∧ proposal[i].state ∈ {ProposalComplete, ProposalFailed} ⇒
            ∀ j ∈ DOMAIN proposal :
              j < i ⇒
                ∧ proposal[j].phase = ProposalCommit ⇒ proposal[j].state = ProposalFailed
                ∧ proposal[j].phase = ProposalApply ⇒ proposal[j].state ∈ {ProposalComplete, ProposalFailed}

IsConsistent(indexes, values) ≜
  LET
    Compute the set of paths in the target that have been updated by transactions
    appliedPaths ≜ UNION {DOMAIN proposal[i].change.values : i ∈ indexes}
    Compute the highest index applied to the target for each path
    pathIndexes ≜ [p ∈ appliedPaths ↦ CHOOSE i ∈ indexes :
                     ∀ j ∈ indexes :
                       ∧ i ≥ j
                       ∧ p ∈ DOMAIN proposal[i].change.values]
    Compute the expected target configuration based on the last indexes applied
    to the target for each path.
    expectedConfig ≜ [p ∈ DOMAIN pathIndexes ↦ proposal[pathIndexes[p]].change.values[p]]
    Compute the actual configuration by converting missing path values to Nil
    actualConfig ≜ [p ∈ DOMAIN expectedConfig ↦ IF p ∈ DOMAIN values THEN values[p] ELSE [value ↦
  IN
    actualConfig ≠ expectedConfig ⇒ ¬(PrintT(indexes) ∧ PrintT(appliedPaths) ∧ PrintT(pathIndexes) ∧ P

Consistency ≜
  ∧ LET indexes ≜ {i ∈ DOMAIN proposal : ∧ ∨ ∧ proposal[i].phase = ProposalCommit
                                             ∧ proposal[i].state  = ProposalComplete
                                          ∨ proposal[i].phase = ProposalApply
                                        ∧ ¬∃ j ∈ DOMAIN proposal :
                                             ∧ j > i
                                             ∧ proposal[j].type = ProposalRollback
                                             ∧ proposal[j].rollback.index = i
                                             ∧ proposal[j].phase = ProposalCommit
                                             ∧ proposal[j].state  = ProposalComplete}
    IN  IsConsistent(indexes, configuration.committed.values)
  ∧ LET indexes ≜ {i ∈ DOMAIN proposal : ∧ proposal[i].phase = ProposalApply
                                         ∧ proposal[i].state  = ProposalComplete
                                         ∧ ¬∃ j ∈ DOMAIN proposal :
                                             ∧ j > i
                                             ∧ proposal[j].type = ProposalRollback
                                             ∧ proposal[j].rollback.index = i
                                             ∧ proposal[j].phase = ProposalApply
                                             ∧ proposal[j].state  = ProposalComplete}
    IN  IsConsistent(indexes, configuration.applied.values)

Safety ≜ □(Order ∧ Consistency)
```

THEOREM $Spec \Rightarrow Safety$

$$
\begin{array}{l}
Terminated(i) \triangleq \\
\quad \land\ i \in \text{DOMAIN } proposal \\
\quad \land\ \lor\ \land\ proposal[i].phase = ProposalApply \\
\qquad\quad \land\ proposal[i].state\ = ProposalComplete \\
\qquad \lor\ proposal[i].state = ProposalFailed
\end{array}
$$

$$
\begin{array}{l}
Termination \triangleq \\
\quad \forall\, i \in 1\,..\,Len(proposal) : \\
\qquad Terminated(i)
\end{array}
$$

$$Liveness \triangleq \Diamond Termination$$

THEOREM $Spec \Rightarrow Liveness$

* Modification History
* Last modified *Fri Apr* 21 18:30:03 *PDT* 2023 by *jhalterm*
* Last modified *Mon Feb* 21 01:32:07 *PST* 2022 by *jordanhalterman*
* Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*