```
1  ---------- MODULE Ledger ----------
   (* High level specification of DLT Ledger, expressed as a single state machine without MVCC validation *)
6  EXTENDS Sequences, Integers, TLAPS, Datatype

   (* State variables of this module *)
11 VARIABLES state,      (* current state of the ledger state machine. *)
12           chain,      (* blockchain, a list of received transactions. *)
13           index       (* unprocessed TX index at the blockchain. *)
14 vars ≜ ⟨state, chain, index⟩

   (* Type invariant *)
19 ChainEntry ≜ [tx : TX, is_valid : BOOLEAN ∪ {NULL}]
20 Chain ≜ Seq(ChainEntry)
21 TypeInv ≜
22     ∧ state ∈ State
23     ∧ index ∈ Nat
24     ∧ index > 0
25     (* Each TX in the blockchain has a flag if it's valid or not. Before the TX
26        is processed, its value is NULL. *)
27     ∧ chain ∈ Chain
28 ----------

   (* Initial condition *)
32 Init ≜
33     ∧ state = InitState      (* state is at the initial state, and *)
34     ∧ index = 1
35     ∧ chain = ⟨⟩             (* empty transaction queue. *)

37 ----------

   (* Actions *)

   (* SubmitTX: Client appends a transaction to the transaction queue. *)
45 SubmitTX(tx) ≜
46     ∧ chain' = Append(chain, [tx ↦ tx, is_valid ↦ NULL])
47     ∧ UNCHANGED ⟨state, index⟩

   (* ProcessTx: Ledger processes the oldest unprocessed TX and updates its state by applyinng f *)
54 ProcessTX_OK ≜
55     LET
56         f ≜ chain[index].tx.f
57     IN
58         (* ∧ Len(chain) ≥ index *)
59         ∧ index ∈ DOMAIN chain
60         ∧ chain' = [chain EXCEPT ![index].is_valid = TRUE]   (* update validity flag *)
```

```
61          ∧ index' = index + 1     increment the index.
62          ∧ state' ∈ f[state]      perform non-deterministic state transition by f.

64  ProcessTX_ERR ≜
65      LET
66          f ≜ chain[index].tx.f
67      IN
68           ∧ Len(chain) ≥ index
69          ∧ index ∈ DOMAIN chain
70          ∧ chain' = [chain EXCEPT ![index].is_valid = FALSE]   see above.
71          ∧ index' = index + 1   see above.
72          ∧ UNCHANGED state      state does not change due to invalid TX.

74  Next ≜ (∃ tx ∈ TX : SubmitTX(tx)) ∨ ProcessTX_OK ∨ ProcessTX_ERR

    Specification
79  Spec ≜ Init ∧ □[Next]_vars

81  ├──────────────────────────────────────────────────────────────────┤
    Invariants
85  Finality ≜ TRUE  TODO
86  Safety ≜ Finality

88   Invariant (safety) on the blockchain
89  ChainInv ≜
90       chain = (processed part) + (unprocessed part)
91      ∧ ∀ i ∈ 1 .. index − 1 : chain[i].is_valid ∈ BOOLEAN
92      ∧ ∀ i ∈ {i ∈ Nat : index ≤ i} ∩ DOMAIN chain : chain[i].is_valid = NULL

94  Inv ≜ TypeInv ∧ ChainInv

96   Invariant (safety) on the high-level Ledger
97  THEOREM LedgerInv ≜ Spec ⇒ □Inv
98  PROOF
99      ⟨1⟩1 Init ⇒ Inv
100         BY InitStateAxiom DEF Init, Inv, TypeInv, ChainInv, Chain
101     ⟨1⟩2 Inv ∧ [Next]_vars ⇒ Inv'
102         ⟨2⟩1 SUFFICES ASSUME TypeInv, ChainInv, [Next]_vars PROVE Inv' BY DEF Inv
103         ⟨2⟩2 CASE Next
104             ⟨3⟩ USE DEF Inv, Next
105             ⟨3⟩ USE DEF TypeInv, ChainInv, Chain, ChainEntry
106             ⟨3⟩1 CASE (∃ tx ∈ TX : SubmitTX(tx))
107                 ⟨4⟩ USE DEF SubmitTX
108                 ⟨4⟩a ∀ i ∈ DOMAIN chain : chain[i] = chain'[i] BY ⟨3⟩1
109                 ⟨4⟩1 TypeInv' BY ⟨2⟩1, ⟨3⟩1
110                 ⟨4⟩2 ChainInv'
111                     ⟨5⟩1 ChainInv!1' OBVIOUS
```

```
                    ⟨5⟩2 ChainInv!2′
                        ⟨6⟩a DOMAIN chain′ = DOMAIN chain ∪ {Len(chain) + 1}BY TypeInv, ⟨3⟩1
                        ⟨6⟩1 PICK tx ∈ TX : SubmitTX(tx)BY ⟨3⟩1
                        ⟨6⟩2 TAKE i ∈ ({i ∈ Nat : index ≤ i} ∩ DOMAIN chain)′
                        ⟨6⟩3CASE i  ∈ ({j ∈ Nat : index ≤ j} ∩ {Len(chain) + 1})BY ⟨2⟩1, ⟨4⟩a, ⟨6⟩1, ⟨6⟩3
                        ⟨6⟩4CASE i  ∈ ({j ∈ Nat : index ≤ j} ∩ DOMAIN chain)BY ⟨2⟩1, ⟨4⟩a, ⟨6⟩1, ⟨6⟩4
                        ⟨6⟩ QED BY ⟨2⟩1, ⟨6⟩a, ⟨6⟩1, ⟨6⟩2, ⟨6⟩3, ⟨6⟩4
                    ⟨5⟩ QED BY ⟨5⟩1, ⟨5⟩2
                ⟨4⟩ QED BY ⟨4⟩1, ⟨4⟩2
            ⟨3⟩2CASE ProcessTX_OK
                ⟨4⟩ USE  DEF ProcessTX_OK
                ⟨4⟩1 TypeInv′BY ⟨2⟩1, ⟨3⟩2  DEF TX, Operation, TotalFunc
                ⟨4⟩2 ChainInv′
                    ⟨5⟩ ChainInv!1′OBVIOUS
                    ⟨5⟩ ChainInv!2′BY ⟨2⟩1, ⟨3⟩2
                    ⟨5⟩ QED OBVIOUS
                ⟨4⟩ QED BY ⟨4⟩1, ⟨4⟩2
            ⟨3⟩3CASE ProcessTX_ERR
                ⟨4⟩ USE  DEF ProcessTX_ERR
                ⟨4⟩1 TypeInv′BY ⟨2⟩1, ⟨2⟩2, ⟨3⟩3  DEF TX, Operation, TotalFunc
                ⟨4⟩2 ChainInv′
                    ⟨5⟩ ChainInv!1′OBVIOUS
                    ⟨5⟩ ChainInv!2′BY ⟨2⟩1, ⟨3⟩3
                    ⟨5⟩ QED OBVIOUS
                ⟨4⟩ QED BY ⟨4⟩1, ⟨4⟩2
            ⟨3⟩ QED
                BY ⟨2⟩1, ⟨2⟩2, ⟨3⟩1, ⟨3⟩2, ⟨3⟩3
        ⟨2⟩3CASE UNCHANGED vars
            BY ⟨2⟩1, ⟨2⟩3  DEF Inv, TypeInv, ChainInv, vars
        ⟨2⟩ QED BY ⟨2⟩1, ⟨2⟩2, ⟨2⟩3
    ⟨1⟩ QED BY PTL, ⟨1⟩1, ⟨1⟩2  DEF Spec

====
\* Modification History
\* Last modified Fri Jul 19 17:59:56 JST 2019 by shinsa
\* Created Fri Jun 07 01:51:28 JST 2019 by shinsa
```