MODULE *Ledger*

High level specification of *DLT Ledger*, expressed as a single state machine without *MVCC* validation

EXTENDS *Sequences*, *Integers*, *TLAPS*

Constants

CONSTANTS *State*, *InitState* a set of states, and

ASSUME *InitStateAxiom* $\triangleq$ *InitState* $\in$ *State* the designated initial state.

$NULL \triangleq$ CHOOSE $x : x \notin$ BOOLEAN

State variables of this module

VARIABLES *state*, current state of the ledger state machine.

chain, blockchain, a list of received transactions.

index index of the blockchain.

$vars \triangleq \langle state, chain, index \rangle$

Datatype definition

$TotalFunc(S1, S2) \triangleq [S1 \to S2 \setminus \{\{\}\}]$ a set of total function from $S1$ to $S2$

Operation is a function from a state to a state, can be non-deterministic, but required to be total.

$Operation \triangleq TotalFunc(State,$ SUBSET $State)$

$Operation \triangleq [State \to$ (SUBSET $State) \setminus \{\{\}\}]$

$TX \triangleq [f : Operation]$ a transaction. note that "f" is just a label

Type invariant

$ChainEntry \triangleq [tx : TX, is\_valid :$ BOOLEAN $\cup \{NULL\}]$

$Chain \triangleq Seq(ChainEntry)$

$TypeInv \triangleq$

$\land\ state \in State$

$\land\ index \in Nat$

$\land\ index > 0$

Each TX in the blockchain has a flag if it's valid or not. Before the TX is processed, its value is $NULL$.

$\land\ chain \in Chain$

Initial condition

$Init \triangleq$

$\land\ state = InitState$ state is at the initial state, and

$\land\ index = 1$

$\land\ chain = \langle\rangle$ empty transaction queue.

Actions

SubmitTX: A client appends a transaction to the transaction queue.

```
66  SubmitTX(tx) ≜
67      ∧ chain' = Append(chain, [tx ↦ tx, is_valid ↦ NULL])
68      ∧ UNCHANGED ⟨state, index⟩

    CommitTx: Ledger processes the oldest unprocessed TX and
73  ProcessTX_OK ≜
74      LET
75          f ≜ chain[index].tx.f
76      IN
77          ∧ Len(chain) ≥ index
78          ∧ index ∈ DOMAIN chain
79          ∧ chain' = [chain EXCEPT ![index].is_valid = TRUE]   update validity flag
80          ∧ index' = index + 1    increment the index.
81          ∧ state' ∈ f[state]     perform non-deterministic state transition by f.

83  ProcessTX_ERR ≜
84      LET
85          f ≜ chain[index].tx.f
86      IN
87          ∧ Len(chain) ≥ index
88          ∧ index ∈ DOMAIN chain
89          ∧ chain' = [chain EXCEPT ![index].is_valid = FALSE]   see above.
90          ∧ index' = index + 1    see above.
91          ∧ UNCHANGED state    state does not change due to invalid TX.

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

    Specification
98  Spec ≜ Init ∧ □[Next]_vars

100 ═══════════════════════════════════════════════════════════════

    Invariants
104 Finality ≜ TRUE  TODO
105 Safety ≜ Finality

107  Invariant (safety) on the blockchain
108 ChainInv ≜
109      chain = (processed part) + (unprocessed part)
110     ∧ ∀ i ∈ 1 .. index − 1 : chain[i].is_valid ∈ BOOLEAN
111     ∧ ∀ i ∈ {i ∈ Nat : index ≤ i} ∩ DOMAIN chain : chain[i].is_valid = NULL

113 Inv ≜ TypeInv ∧ ChainInv

115 THEOREM LedgerInv ≜ Spec ⇒ □Inv
116 PROOF
117     ⟨1⟩1 Init ⇒ Inv
118         BY InitStateAxiom DEF Init, Inv, TypeInv, ChainInv, Chain
```

```
119     ⟨1⟩2 Inv ∧ [Next]_vars ⇒ Inv′
120         ⟨2⟩1 SUFFICES ASSUME TypeInv, ChainInv, [Next]_vars PROVE Inv′ BY DEF Inv
121         ⟨2⟩2 CASE Next
122             ⟨3⟩ USE DEF Inv, Next
123             ⟨3⟩ USE DEF TypeInv, ChainInv, Chain, ChainEntry
124             ⟨3⟩1 CASE (∃ tx ∈ TX : SubmitTX(tx))
125                 ⟨4⟩ USE DEF SubmitTX
126                 ⟨4⟩a ∀ i ∈ DOMAIN chain : chain[i] = chain′[i] BY ⟨3⟩1
127                 ⟨4⟩1 TypeInv′ BY ⟨2⟩1, ⟨3⟩1
128                 ⟨4⟩2 ChainInv′
129                     ⟨5⟩1 ChainInv!1′ OBVIOUS
130                     ⟨5⟩2 ChainInv!2′
131                         ⟨6⟩a DOMAIN chain′ = DOMAIN chain ∪ {Len(chain) + 1} BY TypeInv, ⟨3⟩1
132                         ⟨6⟩1 PICK tx ∈ TX : SubmitTX(tx) BY ⟨3⟩1
133                         ⟨6⟩2 TAKE i ∈ ({i ∈ Nat : index ≤ i} ∩ DOMAIN chain)′
134                         ⟨6⟩3 CASE i ∈ ({j ∈ Nat : index ≤ j} ∩ {Len(chain) + 1}) BY ⟨2⟩1, ⟨4⟩a, ⟨6⟩1, ⟨6⟩3
135                         ⟨6⟩4 CASE i ∈ ({j ∈ Nat : index ≤ j} ∩ DOMAIN chain) BY ⟨2⟩1, ⟨4⟩a, ⟨6⟩1, ⟨6⟩4
136                         ⟨6⟩ QED BY ⟨2⟩1, ⟨6⟩a, ⟨6⟩1, ⟨6⟩2, ⟨6⟩3, ⟨6⟩4
137                     ⟨5⟩ QED BY ⟨5⟩1, ⟨5⟩2
138                 ⟨4⟩ QED BY ⟨4⟩1, ⟨4⟩2
139             ⟨3⟩2 CASE ProcessTX_OK
140                 ⟨4⟩ USE DEF ProcessTX_OK
141                 ⟨4⟩1 TypeInv′ BY ⟨2⟩1, ⟨3⟩2 DEF TX, Operation, TotalFunc
142                 ⟨4⟩2 ChainInv′
143                     ⟨5⟩ ChainInv!1′ OBVIOUS
144                     ⟨5⟩ ChainInv!2′ BY ⟨2⟩1, ⟨3⟩2
145                     ⟨5⟩ QED OBVIOUS
146                 ⟨4⟩ QED BY ⟨4⟩1, ⟨4⟩2
147             ⟨3⟩3 CASE ProcessTX_ERR
148                 ⟨4⟩ USE DEF ProcessTX_ERR
149                 ⟨4⟩1 TypeInv′ BY ⟨2⟩1, ⟨2⟩2, ⟨3⟩3 DEF TX, Operation, TotalFunc
150                 ⟨4⟩2 ChainInv′
151                     ⟨5⟩ ChainInv!1′ OBVIOUS
152                     ⟨5⟩ ChainInv!2′ BY ⟨2⟩1, ⟨3⟩3
153                     ⟨5⟩ QED OBVIOUS
154                 ⟨4⟩ QED BY ⟨4⟩1, ⟨4⟩2
155             ⟨3⟩ QED
156                 BY ⟨2⟩1, ⟨2⟩2, ⟨3⟩1, ⟨3⟩2, ⟨3⟩3
157         ⟨2⟩3 CASE UNCHANGED vars
158             BY ⟨2⟩1, ⟨2⟩3 DEF Inv, TypeInv, ChainInv, vars
159         ⟨2⟩ QED BY ⟨2⟩1, ⟨2⟩2, ⟨2⟩3
160     ⟨1⟩ QED BY PTL, ⟨1⟩1, ⟨1⟩2 DEF Spec

162 ═══════════════════════════════════════════════════════════════════════
    \* Modification History
```

\ * Last modified *Wed Jul* 10 01:23:37 *JST* 2019 by *shinsa*
\ * Created *Fri Jun* 07 01:51:28 *JST* 2019 by *shinsa*