Property 15.16 (Stake Pool Reaping).

$$\begin{aligned}
&\forall l, l' \in \mathsf{ledgerState}, e \in \mathsf{Epoch} : \mathsf{validLedgerstate}\ l, \\
&\qquad l = (_, (d, p)), l' = (_, (d', p')), pp \in \mathsf{PParams}, acnt, acnt' \in \mathsf{Acnt} \\
&\implies pp \vdash (acnt, d, p) \xrightarrow[\text{POOLREAP}]{e} (acnt, d', p') \implies \forall retire \in (\mathsf{getRetiring}\ p)^{-1}[e], retire \neq \emptyset \\
&\qquad \wedge\ retire \subseteq \mathsf{dom}(\mathsf{getStPool}\ p) \wedge retire \cap \mathsf{dom}(\mathsf{getStPool}\ p') = \emptyset \\
&\qquad\qquad \wedge\ retire \cap \mathsf{dom}(\mathsf{getRetiring}\ p') = \emptyset
\end{aligned}$$

Property 15.16 states that for l, l' as above but with a delegation transition of type POOLREAP, there exist registered stake pools in l which are associated to stake pool registration certificates and which are to be retired at the current epoch e. In l' all those stake pools are removed from the maps $stpools$ and $retiring$.

15.6 Properties of Numerical Calculations

The numerical calculations for refunds and rewards in (see Section 11) are also required to have certain properties. In particular we need to make sure that the functions that use non-integral arithmetic have properties which guarantee consistency of the system. Here, we state those properties and formulate them in a way that makes them usable in properties-based testing for validation in the executable spec.

Property 15.17 (Maximal Pool Reward). The maximal pool reward is the expected maximal reward paid to a stake pool. The sum of all these rewards cannot exceed the total available reward, let $Pool$ be the set of active stake pools:

$$\forall R \in Coin : \sum_{p \in Pools} \left\lfloor \frac{R}{1 + p_{a_0}} \cdot \left(p_{\sigma'} + p_{p' \cdot a_0} \cdot \frac{p_{\sigma'} - p_{p'} \cdot \frac{p_{z_0} - p_{\sigma'}}{p_{z_0}}}{p_{z_0}} \right) \right\rfloor \leq R$$

Property 15.18 (Actual Reward). The actual reward for a stake pool in an epoch is calculated by the function poolReward. The actual reward per stake pool is non-negative and bounded by the maximal reward for the stake pool, with $\overline{p}$ being the relation $\frac{n}{\max(1, \overline{N})}$ of the number of produced blocks n of one pool to the total number $\overline{N}$ of produced blocks in an epoch and $maxP$ being the maximal reward for the stake pool. This gives us:

$$\forall \gamma \in [0, 1] \implies 0 \leq \lfloor \overline{p} \cdot maxP \rfloor \leq maxP$$

The two functions $\mathsf{r}_{\mathsf{operator}}$ and $\mathsf{r}_{\mathsf{member}}$ are closely related as they both split the reward between the pool leader and the members.

Property 15.19 (Reward Splitting). The reward splitting is done via $\mathsf{r}_{\mathsf{operator}}$ and $\mathsf{r}_{\mathsf{member}}$, i.e., a split between the pool leader and the pool members using the pool cost c and the pool margin m. Therefore the property relates the total reward $\hat{f}$ to the split rewards in the following way:

$$\forall m \in [0, 1], c \in Coin \implies c + \left\lfloor (\hat{f} - c) \cdot (m + (1 - m)) \cdot \frac{s}{\sigma} \right\rfloor + \sum_j \left\lfloor (\hat{f} - c) \cdot (1 - m) \cdot \frac{t_j}{\sigma} \right\rfloor \leq \hat{f}$$

15.7 Preservation of Value

As visualized in Figure 49, the total amount of lovelace in any given chain state $s \in \mathsf{ChainState}$ is completely contained within the values of the six variables:

Variable	Name in Figure 49	Nesting Inside Chain State	Kind
utxo	circulation	s.nes.es.ls.utxoSt	Map over Lovelace Values
deposits	deposits	s.nes.es.ls.utxoSt	Lovelace Value (Coin)
fees	fees	s.nes.es.ls.utxoSt	Lovelace Value (Coin)
rewards	reward accounts	s.nes.es.ls.dpstate.dstate	Lovelace Value (Coin)
treasury	treasury	s.nes.es.acnt	Lovelace Value (Coin)
reserves	reserves	s.nes.es.acnt	Map over Lovelace Values

Notice that *deposits*, *fees*, *treasury*, and *reserves* are all single lovelace values, while *utxo*, and *rewards* are maps whose values are lovelace.

We define the *Lovelace Value* of a given chain state as:

Definition 15.2 (Lovelace Value)**.**

$$\mathsf{Val}(s \in \mathit{State}) = \mathsf{Val}(\mathit{utxo}) + \mathsf{Val}(\mathit{deposits}) + \mathsf{Val}(\mathit{fees}) + \mathsf{Val}(\mathit{reserves}) + \mathsf{Val}(\mathit{treasury}) + \mathsf{Val}(\mathit{rewards})$$

where

$$\mathsf{Val}(x \in \mathsf{Coin}) = x$$
$$\mathsf{Val}((\_ \mapsto (y \in \mathsf{Coin}))^*) = \sum y$$

For any state that is used in a given subtransition of CHAIN, we define Val in an analogous way, setting the value of any variable that is not explicitly represented in the state to zero. For example, given $\mathit{utxoSt} \in \mathsf{UTxOState}$,

$$\mathsf{Val}(\mathit{utxoSt}) = \left(\sum_{\_ \mapsto (\_,\, v) \in \mathit{utxo}} v \right) + \mathit{deposits} + \mathit{fees}$$

The key property that we want to prove is that no semantic transition changes the value that is captured in the state (Vals). This property is easy to state: intuitively, the *Lovelace Value*before the transition is the same as the *Lovelace Value* after that transition.

Theorem 15.3 (Preservation of Value)**.** For all environments e, blocks b, and states s, s', if

$$e \vdash s \xrightarrow[\text{CHAIN}]{b} s'$$

then

$$\mathsf{Val}(s) = \mathsf{Val}(s')$$

We will prove the soundness of Theorem 15.3 via a few lemmas.

Lemma 15.4. For any mapping $m : A \mapsto \mathsf{Coin}$ and set $s \in \mathbb{P}\, A$,

$$\mathsf{Val}(m) = \mathsf{Val}(s \not\lhd m) + \mathsf{Val}(s \lhd m)$$

Proof. easy □

Lemma 15.5. For any mappings $m_1, m_2 : A \mapsto \mathsf{Coin}$, if $\operatorname{dom} m_1 \cap \operatorname{dom} m_2 = \emptyset$, then

$$\mathsf{Val}(m_1 \cup m_2) = \mathsf{Val}(m_1) + \mathsf{Val}(m_2)$$

Proof. easy □

Lemma 15.6. For all environments e, transactions t, and states s, s', if

$$e \vdash s \xrightarrow[\text{UTXO}]{t} s'$$

then

$$\mathsf{Val}(s) + w = \mathsf{Val}(s')$$

where $w = \mathsf{wbalance}\ (\mathsf{txwdrls}\ t)$.

Proof. The proof is essentially unfolding the definition of the predicate

$$\mathsf{consumed}\ pp\ utxot = \mathsf{produced}\ pp\ stpoolst \tag{46}$$

and applying a little algebra. If we let:

$$\begin{aligned} k &= \mathsf{keyRefunds}\ pp\ stkCredst \\ f &= \mathsf{txfee}\ t \\ d &= \mathsf{totalDeposits}\ pp\ stpools\ (\mathsf{txcerts}\ t) \end{aligned}$$

then equation 46 can be rewritten as:

$$\mathsf{Val}(\mathsf{txins}\ t \lhd utxo) + w + k = \mathsf{Val}(\mathsf{outs}\ t) + f + d$$

where outs is defined in Figure 14 and returns a value of type UTxO. Therefore, moving k to the right and adding $\mathsf{txins}\ t \lhd utxo$ to each side,

$$\mathsf{Val}(\mathsf{txins}\ t \lhd utxo) + \mathsf{Val}(\mathsf{txins}\ t \lhd utxo) + w = \mathsf{Val}(\mathsf{outs}\ t) + f + d - k + \mathsf{Val}(\mathsf{txins}\ t \lhd utxo)$$

(Though not needed for the proof at hand, note that $d - k$ is non-negative since the deposits will always be large enough to cover the current obligation. See Theorem 15.11.) It then follows that:

$$\begin{aligned} \mathsf{Val}(utxo) + w &= \mathsf{Val}(\mathsf{outs}\ t) + f + d - k + \mathsf{Val}(\mathsf{txins}\ t \lhd utxo) && \text{(by Lemma 15.4)} \\ &= \mathsf{Val}((\mathsf{txins}\ t \lhd utxo) \cup \mathsf{outs}\ t) + (d - k) + f && \text{(by Lemma 15.5)} \end{aligned}$$

Note that in order to apply Lemma 15.5 above, it must be true that $(\mathsf{txins}\ t \lhd utxo)$ and $(\mathsf{outs}\ t)$ have disjoint domains, which follows from the uniqueness of the transaction IDs.

Therefore, by adding the deposits and fees from s to the equality above, it follows that $\mathsf{Val}(s) + w = \mathsf{Val}(s')$. □

Lemma 15.7. For all environments e, transactions c, and states s, s', if

$$e \vdash s \xrightarrow[\text{DELEG}]{c} s'$$

then

$$\mathsf{Val}(s) = \mathsf{Val}(s')$$

Proof. The only variable with value in this transition is *rewards*. Only two of the rules in DELEG can change *rewards*, namely Deleg−Reg and Deleg−Dereg. However, Deleg−Reg only adds a zero value, and Deleg−Dereg only removes a zero value. □

Lemma 15.8. For all environments e, certificates Γ, and states s, s', if

$$e \vdash s \xrightarrow[\text{DELEGS}]{\Gamma} s'$$

then

$$\mathsf{Val}(s) = \mathsf{Val}(s') + w$$

where $w = \mathsf{wbalance}\ (\mathsf{txwdrls}\ t)$, and t is the transaction in the environment e.

Proof. The proof is by induction on the length of Γ. Note that the only variable with value in this transition is *rewards*.

In the base case, we look at the rule Seq−delg−base. Since $wdrls \subseteq rewards$, then $rewards = wdrls \cup (rewards \setminus wdrls)$. Therefore

$$\begin{aligned} \mathsf{Val}(rewards) &= \mathsf{Val}(rewards \setminus wdrls) + \mathsf{Val}(wdrls) && \text{by Lemma 15.5} \\ &= \mathsf{Val}(rewards \setminus wdrls) + w && \text{by definition} \\ &= \mathsf{Val}\left(rewards \underset{\rightarrow}{\cup} \{(w, 0) \mid w \in \operatorname{dom} wdrls\}\right) + w \end{aligned}$$

Therefore $\mathsf{Val}(s) = \mathsf{Val}(s')$.

In the inductive case, we look at the rule Seq−delg−ind. In this case, the lemma then follows directly from Lemma 15.7. □

Lemma 15.9. For all environments e, epoch ϵ, and states s, s', if

$$e \vdash s \xrightarrow[\text{POOLREAP}]{\epsilon} s'$$

then

$$\mathsf{Val}(s) = \mathsf{Val}(s')$$

Proof. The POOLREAP value is contained in *deposits*, *treasury*, and *rewards*. Notice that *unclaimed* is added to *treasury* and subtracted from the *deposits*. Moreover, *refunded* is subtracted from *deposits*. (Note that $deposits - (unclaimed + refunded)$ is non-negative by Theorem 15.11.) It therefore suffices to show that

$$\begin{aligned} \mathsf{Val}(rewards \cup_+ refunds) &= \mathsf{Val}(rewards) + \mathsf{Val}(refunds) \\ &= \mathsf{Val}(rewards) + refunded \end{aligned}$$

But this is clear from the definition of $\cup_+$. □

Lemma 15.10. For every $(\Delta t,\ \Delta r,\ rs,\ \Delta f)$ in the range of createRUpd,

$$\Delta t + \Delta r + \mathsf{Val}(rs) + \Delta f = 0$$

Proof. In the definition of createRUpd in Figure 51, We see that:

$$\begin{aligned} rewardPot &= feeSS + \Delta r \\ R &= rewardPot - \Delta t_1 \\ \Delta t_2 &= R - \mathsf{Val}(rs) \\ \Delta t &= \Delta t_1 + \Delta t_2 \end{aligned}$$

Therefore

$$\begin{aligned} (feeSS + \Delta r) &= rewardPot = R + \Delta t_1 = \Delta t_2 + \mathsf{Val}(rs) + \Delta t_1 \\ 0 &= (\Delta t_1 + \Delta t_2) - \Delta r + \mathsf{Val}(rs) - feeSS \\ 0 &= \Delta t - \Delta r + \mathsf{Val}(rs) - feeSS \end{aligned}$$

It then suffices to notice that createRUpd returns $(\Delta t, -\Delta r,\ rs,\ -feeSS)$. □

Note that Lemma 15.10 is not strictly need for the proof of Theorem 15.3, since the NEWEPOCH transition requires that $\Delta t + \Delta r + \mathsf{Val}(rs) + \Delta f = 0$ holds. It does, however, give us confidence that the CHAIN transition can proceed.

We are now ready to prove Theorem 15.3.

Proof. For a given transition TR, let PresOfVal(TR) be the statement:

$$\text{for all environments } e, \text{ signals } \sigma, \text{ and states } s, s',\ e \vdash s \xrightarrow[\text{TR}]{\sigma} s' \implies \mathsf{Val}(s) = \mathsf{Val}(s').$$

Our goal is to prove PresOfVal(CHAIN). Lemmas 15.6 and 15.8 imply PresOfVal(LEDGER), since UTXOW transforms state exactly as UTXO does. PresOfVal(LEDGERS) then follows by straightforward induction on the length of Γ: the base case is trivial; and the inductive case follows directly from PresOfVal(LEDGER). PresOfVal(SNAP) holds trivially, since it contains no value. Similarly, PresOfVal(NEWPP) holds since *diff* is added to *reserves* and subtracted from *deposits*. Therefore PresOfVal(EPOCH) holds by Lemma 15.9. PresOfVal(MIR) holds since $\mathsf{Val}i'_{rwd} = tot$ in Figure 55. Morover, PresOfVal(NEWEPOCH) holds in the presence of applyRUpd since the transition requires $\Delta t + \Delta r + \mathsf{Val}(rs) + \Delta f = 0$. PresOfVal(CHAIN) easily follows from this. □

15.8 Non-negative Deposit Pot

The *deposit pot* (the variable *deposits* in the UTxO State) represents the amount of *lovelace* that is set aside by the system as a whole for refunding deposits. Deposits are added to this pot, which then decays exponentially over time, and is also depleted by any refunded deposits. At an epoch boundary, the decayed parts of any deposits (including, possibly, deposits for any transactions that will complete in future epochs) will be distributed as additional *rewards*, as described in [SL-D1]. Since *deposits* is only used to record the value of future refunds or rewards whose costs have already been incurred, both it and any reward value will always be non-negative. Note that there are two types of deposits which are recorded in the same pot: those for stake keys; and those for stake pools. Stake keys are deregistered in the slot in which the deregistration certificates is processed. Stake pools, however, are staged for retirement on epoch boundaries. The following theorem ensures that the deposit pot is properly maintained and will always be large enough to meet all of its obligations.

Variable	Value	Nesting Inside Chain State	Kind
deposits	0	s.nes.es.ls.utxoSt	Coin
stkCreds	∅	s.nes.es.ls.dpstate.dstate.stkCreds	StakeCreds (Credential ↦ Slot)
stpools	∅	s.nes.es.ls.dpstate.pstate.stpools	StakePools (KeyHash ↦ Slot)

Figure 83: Initial Chain State

Theorem 15.11 (Non-negative Deposit Pot)**.** Let $n \in \mathbb{N}$ and $c_0 \in$ ChainState be a chain state in which $deposits = 0$, $stkCreds = \emptyset$ and $stPools = \emptyset$, as shown above: If

$$s_0 \vdash c_0 \xrightarrow[\text{CHAIN}]{b_0} c_1,\ \ s_1 \vdash c_1 \xrightarrow[\text{CHAIN}]{b_1} c_2,\ \ \ldots,\ \ s_n \vdash c_n \xrightarrow[\text{CHAIN}]{b_n} c_{n+1},\ \ n \geq 0$$

is a sequence of valid CHAIN transitions, then $\forall i, 0 \leq i \leq n, deposits\ (c_{n+1}) \geq 0$.

Proof. We will prove a slightly stronger condition, namely that some stronger invariants hold most of the time, and that when they do fail to hold, then *deposits* is still non-negative. These stronger invariants will require a few additional definitions. Given a slot s, let $\ell(s)$ be the first slot of the epoch that s occurs in, that is $\ell = \mathsf{firstSlot} \circ \mathsf{epoch}$. Given a mapping $m \in \mathsf{T} \to \mathsf{Slot}$ and a slot $s \in \mathsf{Slot}$, let sep be the function that separates m into two maps, those whose value is strictly less than s and those whose value is at least s. So,

$$\mathsf{sep}\ m\ s = \forall x \mapsto t\ \in\ m,\ \ (\{x \mapsto t\ \mid\ t < s\},\ \{x \mapsto t\ \mid\ t \geq s\})$$