Figure 51 defines two functions, createRUpd to create a reward update and applyRUpd to apply a reward update to an instance of EpochState.

The createRUpd function does the following:

- Note that for all the calculations below, we use the previous protocol parameters *prevPp*, which corresponds to the parameters during the epoch for which we are creating rewards.
- First we calculate the change to the reserves, as determined by the ρ protocol parameter.
- Next we calculate *rewardPot*, the total amount of coin available for rewards this epoch, as described in section 6.4 of [SL-D1]. It consists of:
  - The fee pot, containing the transaction fees from the epoch.
  - The amount of monetary expansion from the reserves, calculated above.

  Note that the fee pot is taken from the snapshot taken at the epoch boundary. (See Figure39).
- Next we calculate the proportion of the reward pot that will move to the treasury, as determined by the τ protocol parameter. The remaining pot is called the R, just as in section 6.5 of [SL-D1].
- The rewards are calculated, using the oldest stake distribution snapshot (the one labeled "go"). As given by maxPool, each pool can receive a maximal amount, determined by its performance. The difference between the maximal amount and the actual amount received is added to the amount moved to the treasury.
- The fee pot will be reduced by *feeSS*.

Note that fees are not explicitly removed from any account: the fees come from transactions paying them and are accounted for whenever transactions are processed.

The applyRUpd function does the following:

- Adjust the treasury, reserves and fee pots by the appropriate amounts.
- Add each individual reward to the global reward mapping. We must be careful, though, not to give out rewards to accounts that have been deregistered after the reward update was created.
  - Rewards for accounts that are still registered are added to the reward mappings.
  - The sum of the unregistered rewards are added to the reserves.

These two functions will be used in the blockchain transition systems in Section 12. In particular, createRUpd will be used in Equation 34, and applyRUpd will be used in Equation 27.

Calculation to create a reward update

$$\mathsf{createRUpd} \in \mathbb{N} \to \mathsf{BlocksMade} \to \mathsf{EpochState} \to \mathsf{Coin} \to \mathsf{RewardUpdate}$$

$$\mathsf{createRUpd}\ \mathit{slotsPerEpoch}\ b\ \mathit{es}\ \mathit{total} = (\Delta t_1, -\Delta r_1 + \Delta r_2,\ \mathit{rs},\ -\mathit{feeSS})$$

where

$$(\mathit{acnt},\ \mathit{ss},\ \mathit{ls},\ \mathit{prevPp},\ \_) = \mathit{es}$$

$$(\_,\ \_,\ \mathit{pstake}_{go},\ \mathit{feeSS}) = \mathit{ss}$$

$$(\mathit{stake},\ \mathit{delegs},\ \mathit{poolParams}) = \mathit{pstate}_{go}$$

$$(\_,\ \mathit{reserves}) = \mathit{acnt}$$

$$(\_,\ ((\mathit{rewards},\ \_,\ \_,\ \_,\ \_,\ \_)\ ,\ \_)) = \mathit{ls}$$

$$\Delta r_1 = \lfloor \min(1, \eta) \cdot (\mathsf{rho}\ \mathit{prevPp}) \cdot \mathit{reserves} \rfloor$$

$$\eta = \begin{cases} 1 & (\mathsf{d}\ \mathit{prevPp}) \geq 0.8 \\ \frac{\mathit{blocksMade}}{\lceil (1 - \mathsf{d}\ \mathit{prevPp}) \cdot \mathit{slotsPerEpoch} \cdot \mathsf{ActiveSlotCoeff} \rceil} & \text{otherwise} \end{cases}$$

$$\mathit{rewardPot} = \mathit{feeSS} + \Delta r_1$$

$$\Delta t_1 = \lfloor (\mathsf{tau}\ \mathit{prevPp}) \cdot \mathit{rewardPot} \rfloor$$

$$R = \mathit{rewardPot} - \Delta t_1$$

$$\mathit{circulation} = \mathit{total} - \mathit{reserves}$$

$$\mathit{rs} = \mathsf{reward}\ \mathit{prevPp}\ b\ R\ (\mathrm{dom}\ \mathit{rewards})\ \mathit{poolParams}\ \mathit{stake}\ \mathit{delegs}\ \mathit{circulation}$$

$$\Delta r_2 = R - \left(\sum_{\_ \mapsto c \in \mathit{rs}} c \right)$$

$$\mathit{blocksMade} = \sum_{\_ \mapsto m \in b} m$$

Figure 51: Reward Update Creation

Applying a reward update

$$\mathsf{applyRUpd} \in \mathsf{RewardUpdate} \to \mathsf{EpochState} \to \mathsf{EpochState}$$

$$\mathsf{applyRUpd} \begin{pmatrix} \Delta t \\ \Delta r \\ rs \\ \Delta f \end{pmatrix} \begin{pmatrix} treasury \\ reserves \\ \\ rewards \\ delegations \\ ptrs \\ genDelegs \\ fGenDelegs \\ i_{rwd} \\ \\ poolParams \\ fPoolParams \\ retiring \\ \\ utxo \\ deposited \\ fees \\ up \\ \\ prevPp \\ pp \end{pmatrix} = \begin{pmatrix} \mathbf{treasury + \Delta t + unregRU'} \\ \mathbf{reserves + \Delta r} \\ \\ \mathbf{rewards \cup_{+} regRU} \\ delegations \\ ptrs \\ genDelegs \\ fGenDelegs \\ i_{rwd} \\ \\ poolParams \\ fPoolParams \\ retiring \\ \\ utxo \\ deposited \\ \mathbf{fees + \Delta f} \\ up \\ \\ prevPp \\ pp \end{pmatrix}$$

where

$$regRU = (\operatorname{dom} rewards) \lhd rs$$

$$unregRU = (\operatorname{dom} rewards) \not\lhd rs$$

$$unregRU' = \sum_{\_ \mapsto c \in unregRU} c$$

Figure 52: Reward Update Application

12 Blockchain layer

This chapter introduces the view of the blockchain layer as required for the ledger. This includes in particular the information required for the epoch boundary and its rewards calculation as described in Section 11. It also covers the transitions that keep track of produced blocks in order to calculate rewards and penalties for stake pools.

The main transition rule is CHAIN which calls the subrules NEWEPOCH and UPDN, VRF and BBODY.

12.1 Block Definitions

Abstract types

$$\begin{array}{rcll} h & \in & \mathsf{HashHeader} & \text{hash of a block header} \\ hb & \in & \mathsf{HashBBody} & \text{hash of a block body} \\ bn & \in & \mathsf{BlockNo} & \text{block number} \end{array}$$

Operational Certificate

$$\mathsf{OCert} = \left(\begin{array}{rcll} vk_{hot} & \in & \mathsf{VKey_{ev}} & \text{operational (hot) key} \\ n & \in & \mathbb{N} & \text{certificate issue number} \\ c_0 & \in & \mathsf{KESPeriod} & \text{start KES period} \\ \sigma & \in & \mathsf{Sig} & \text{cold key signature} \end{array}\right)$$

Block Header Body

$$\mathsf{BHBody} = \left(\begin{array}{rcll} prev & \in & \mathsf{HashHeader}^? & \text{hash of previous block header} \\ vk & \in & \mathsf{VKey} & \text{block issuer} \\ vrfVk & \in & \mathsf{VKey} & \text{VRF verification key} \\ blockno & \in & \mathsf{BlockNo} & \text{block number} \\ slot & \in & \mathsf{Slot} & \text{block slot} \\ \eta & \in & \mathsf{Seed} & \text{nonce} \\ prf_\eta & \in & \mathsf{Proof} & \text{nonce proof} \\ \ell & \in & [0,\,1] & \text{leader election value} \\ prf_\ell & \in & \mathsf{Proof} & \text{leader election proof} \\ bsize & \in & \mathbb{N} & \text{size of the block body} \\ bhash & \in & \mathsf{HashBBody} & \text{block body hash} \\ oc & \in & \mathsf{OCert} & \text{operational certificate} \\ pv & \in & \mathsf{ProtVer} & \text{protocol version} \end{array}\right)$$

Block Types

$$\begin{array}{rcll} bh & \in & \mathsf{BHeader} & = & \mathsf{BHBody} \times \mathsf{Sig} \\ b & \in & \mathsf{Block} & = & \mathsf{BHeader} \times \mathsf{Tx}^* \end{array}$$

Abstract functions

$$\begin{array}{rcll} \mathsf{bhHash} & \in & \mathsf{BHeader} \to \mathsf{HashHeader} & \text{hash of a block header} \\ \mathsf{bHeaderSize} & \in & \mathsf{BHeader} \to \mathbb{N} & \text{size of a block header} \\ \mathsf{bBodySize} & \in & \mathsf{Tx}^* \to \mathbb{N} & \text{size of a block body} \\ \mathsf{slotToSeed} & \in & \mathsf{Slot} \to \mathsf{Seed} & \text{convert a slot to a seed} \\ \mathsf{prevHashToNonce} & \in & \mathsf{HashHeader}^? \to \mathsf{Seed} & \text{convert an optional header hash to a seed} \\ \mathsf{bbodyhash} & \in & \mathsf{Tx}^* \to \mathsf{HashBBody} & \end{array}$$

Accessor Functions

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{bheader} & \in & \mathsf{Block} \to \mathsf{BHeader} & \mathsf{bhbody} & \in & \mathsf{BHeader} \to \mathsf{BHBody} \\ \mathsf{hsig} & \in & \mathsf{BHeader} \to \mathsf{Sig} & \mathsf{bbody} & \in & \mathsf{Block} \to \mathsf{Tx}^* \\ \mathsf{bvkcold} & \in & \mathsf{BHBody} \to \mathsf{VKey} & \mathsf{bvkvrf} & \in & \mathsf{BHBody} \to \mathsf{VKey} \\ \mathsf{bprev} & \in & \mathsf{BHBody} \to \mathsf{HashHeader}^? & \mathsf{bslot} & \in & \mathsf{BHBody} \to \mathsf{Slot} \\ \mathsf{bblockno} & \in & \mathsf{BHBody} \to \mathsf{BlockNo} & \mathsf{bnonce} & \in & \mathsf{BHBody} \to \mathsf{Seed} \\ \mathsf{bprf_n} & \in & \mathsf{BHBody} \to \mathsf{Proof} & \mathsf{bleader} & \in & \mathsf{BHBody} \to \mathbb{N} \\ \mathsf{bprf}_\ell & \in & \mathsf{BHBody} \to \mathsf{Proof} & \mathsf{hBbsize} & \in & \mathsf{BHBody} \to \mathbb{N} \\ \mathsf{bhash} & \in & \mathsf{BHBody} \to \mathsf{HashBBody} & \mathsf{bocert} & \in & \mathsf{BHBody} \to \mathsf{OCert} \end{array}$$

Figure 53: Block Definitions