$$\text{Pool-Reg}\ \frac{\begin{array}{c} c \in \mathsf{DCert}_{\mathsf{regpool}} \qquad hk := \mathsf{cwitness}\ c \qquad pool := \mathsf{poolParam}\ c \\ hk \notin \mathrm{dom}\ poolParams \qquad \mathsf{poolCost}\ pool \geq \mathsf{minPoolCost}\ pp \end{array}}{slot,\ pp \vdash \begin{pmatrix} poolParams \\ fPoolParams \\ retiring \end{pmatrix} \xrightarrow[\text{POOL}]{c} \begin{pmatrix} \boldsymbol{poolParams} & \boldsymbol{\cup} & \boldsymbol{\{hk \mapsto pool\}} \\ fPoolParams & & \\ retiring & & \end{pmatrix}} \tag{11}$$

$$\text{Pool-reReg}\ \frac{\begin{array}{c} c \in \mathsf{DCert}_{\mathsf{regpool}} \qquad hk := \mathsf{cwitness}\ c \qquad pool := \mathsf{poolParam}\ c \\ hk \in \mathrm{dom}\ poolParams \qquad \mathsf{poolCost}\ pool \geq \mathsf{minPoolCost}\ pp \end{array}}{slot,\ pp \vdash \begin{pmatrix} poolParams \\ fPoolParams \\ retiring \end{pmatrix} \xrightarrow[\text{POOL}]{c} \begin{pmatrix} poolParams & & \\ \boldsymbol{fPoolParams} & \boldsymbol{\underset{\rightarrow}{\cup}} & \boldsymbol{\{hk \mapsto pool\}} \\ \boldsymbol{\{hk\}} & \boldsymbol{\lhd} & \boldsymbol{retiring} \end{pmatrix}} \tag{12}$$

$$\text{Pool-Retire}\ \frac{\begin{array}{c} c \in \mathsf{DCert}_{\mathsf{retirepool}} \qquad hk := \mathsf{cwitness}\ c \qquad hk \in \mathrm{dom}\ poolParams \\ e := \mathsf{retire}\ c \qquad cepoch := \mathsf{epoch}\ slot \qquad cepoch < e \leq cepoch + (\mathsf{emax}\ pp) \end{array}}{slot,\ pp \vdash \begin{pmatrix} poolParams \\ fPoolParams \\ retiring \end{pmatrix} \xrightarrow[\text{POOL}]{c} \begin{pmatrix} poolParams & & \\ fPoolParams & & \\ \boldsymbol{retiring} & \boldsymbol{\underset{\rightarrow}{\cup}} & \boldsymbol{\{hk \mapsto e\}} \end{pmatrix}} \tag{13}$$

Figure 25: Pool Inference Rule

9.5 Delegation and Pool Combined Rules

We now combine the delegation and pool transition systems. Figure 26 gives the state, environment and transition type for the combined transition.

Delegation and Pool Combined Environment

$$\mathsf{DPEnv} = \left(\begin{array}{rcll} \mathit{slot} & \in & \mathsf{Slot} & \text{slot} \\ \mathit{ptr} & \in & \mathsf{Ptr} & \text{certificate pointer} \\ \mathit{pp} & \in & \mathsf{PParams} & \text{protocol parameters} \\ \mathit{acnt} & \in & \mathsf{Acnt} & \text{accounting state} \end{array} \right)$$

Delegation and Pool Combined State

$$\mathsf{DPState} = \left(\begin{array}{rcll} \mathit{dstate} & \in & \mathsf{DState} & \text{delegation state} \\ \mathit{pstate} & \in & \mathsf{PState} & \text{pool state} \end{array} \right)$$

Delegation and Pool Combined Transition

$$\_ \vdash \_ \xrightarrow[\text{DELPL}]{} \_ \in \mathbb{P}\left(\mathsf{DPEnv} \times \mathsf{DPState} \times \mathsf{DCert} \times \mathsf{DPState}\right)$$

Figure 26: Delegation and Pool Combined Transition Type

Figure 27, gives the rules for the combined transition. Note that for any given certificate, at most one of the two rules (Equation (14) and Equation (15)) will be successful, since the pool certificates are disjoint from the delegation certificates.

Delegation and Pool Combined Rules

$$\text{Delpl-Deleg}\ \frac{\begin{matrix}slot\\ptr\\acnt\end{matrix} \vdash dstate \xrightarrow[\text{DELEG}]{c} dstate'}{\begin{matrix}slot\\ptr\\pp\\acnt\end{matrix} \vdash \begin{pmatrix} dstate \\ pstate \end{pmatrix} \xrightarrow[\text{DELPL}]{c} \begin{pmatrix} \mathbf{dstate'} \\ pstate \end{pmatrix}} \tag{14}$$

$$\text{Delpl-Pool}\ \frac{\begin{matrix}slot\\pp\end{matrix} \vdash pstate \xrightarrow[\text{POOL}]{c} pstate'}{\begin{matrix}slot\\ptr\\pp\\acnt\end{matrix} \vdash \begin{pmatrix} dstate \\ pstate \end{pmatrix} \xrightarrow[\text{DELPL}]{c} \begin{pmatrix} dstate \\ \mathbf{pstate'} \end{pmatrix}} \tag{15}$$

Figure 27: Delegation and Pool Combined Transition Rules

We now describe a transition system that processes the list of certificates inside a transaction. It is defined recursively from the transition system in Figure 27 above.

Figure 28 defines the types for the delegation certificate sequence transition.

Certificate Sequence Environment

$$\mathsf{DPSEnv} = \begin{pmatrix} slot & \in & \mathsf{Slot} & \text{slot} \\ txIx & \in & \mathsf{Ix} & \text{transaction index} \\ pp & \in & \mathsf{PParams} & \text{protocol parameters} \\ tx & \in & \mathsf{Tx} & \text{transaction} \\ acnt & \in & \mathsf{Acnt} & \text{accounting state} \end{pmatrix}$$

$$\_ \vdash \_ \xrightarrow[\text{DELEGS}]{} \_ \in \mathbb{P}\ (\mathsf{DPSEnv} \times \mathsf{DPState} \times \mathsf{DCert}^* \times \mathsf{DPState})$$

Figure 28: Delegation sequence transition type

Figure 29 defines the transition system recursively. This definition guarantees that a certificate list (and therefore, the transaction carrying it) cannot be processed unless every certificate in it is valid. For example, if a transaction is carrying a certificate that schedules a pool retirement in a past epoch, the whole transaction will be invalid.

- The base case, when the list is empty, is captured by Equation (16). In the base case we address one final accounting detail not yet covered by the UTxO transition, namely setting the reward account balance to zero for any account that made a withdrawal. There is therefore a precondition that all withdrawals are correct, where correct means that there is a reward account for each stake credential and that the balance matches that of the reward being withdrawn. The base case triggers the following state transformation:
  - Reward accounts are set to zero for each corresponding withdrawal.

- The inductive case, when the list is non-empty, is captured by Equation (17). It constructs a certificate pointer given the current slot and transaction index, calls DELPL on the next certificate in the list and inductively calls DELEGS on the rest of the list. The inductive case triggers the following state transformation:
  - The delegation and pool states are (inductively) updated by the results of DELEGS, which is then updated according to DELPL.

$$\text{Seq-delg-base}\ \frac{\begin{array}{c} wdrls := \mathsf{txwdrls}\ (\mathsf{txbody}\ tx) \qquad wdrls \subseteq rewards \\ rewards' := rewards \underset{\rightarrow}{\cup} \{(w, 0) \mid w \in \mathrm{dom}\ wdrls\} \end{array}}{\begin{array}{c} slot \\ txIx \\ pp \\ tx \\ acnt \end{array} \vdash \left(\begin{array}{c} \left(\begin{array}{r} rewards \\ delegations \\ ptrs \\ fGenDelegs \\ genDelegs \\ i_{rwd} \end{array} \right) \\ \left(\begin{array}{c} poolParams \\ fPoolParams \\ retiring \end{array} \right) \end{array} \right) \xrightarrow[\text{DELEGS}]{\epsilon} \left(\begin{array}{c} \left(\begin{array}{r} \boldsymbol{rewards'} \\ delegations \\ ptrs \\ fGenDelegs \\ genDelegs \\ i_{rwd} \end{array} \right) \\ \left(\begin{array}{c} poolParams \\ fPoolParams \\ retiring \end{array} \right) \end{array} \right)} \tag{16}$$

$$\text{Seq-delg-ind}\ \frac{\begin{array}{c} \begin{array}{c} slot \\ txIx \\ pp \\ tx \\ acnt \end{array} \vdash dpstate \xrightarrow[\text{DELEGS}]{\Gamma} dpstate' \\ c \in \mathsf{DCert}_{\mathsf{delegate}} \Rightarrow \mathsf{dpool}\ c \in \mathrm{dom}\ poolParams \\ ptr := (slot,\ txIx,\ \mathsf{len}\ \Gamma) \\ \begin{array}{c} slot \\ ptr \\ pp \\ acnt \end{array} \vdash dpstate' \xrightarrow[\text{DELPL}]{c} dpstate'' \end{array}}{\begin{array}{c} slot \\ txIx \\ pp \\ tx \\ acnt \end{array} \vdash dpstate \xrightarrow[\text{DELEGS}]{\Gamma ; c} \boldsymbol{dpstate''}} \tag{17}$$

Figure 29: Delegation sequence rules

The DELEGS rule has two predicate failures:

- In the case of a key delegation certificate, if the pool key is not registered, there is a *DelegateeNotRegistered* failure.
- If the withdrawals mapping of the transaction is not a subset of the rewards mapping of the delegation state, there is a *WithdrawalsNotInRewards* failure.

10 Ledger State Transition

The entire state transformation of the ledger state caused by a valid transaction can now be given as the combination of the UTxO transition and the delegation transitions.

Figure 30 defines the types for this transition. The environment for this rule consists of:

- The current slot.
- The transaction index within the current block.
- The protocol parameters.
- The accounting state.

The ledger state consists of:

- The UTxO state.
- The delegation and pool states.

Ledger environment

$$\mathsf{LEnv} = \left(\begin{array}{rcll} \mathit{slot} & \in & \mathsf{Slot} & \text{current slot} \\ \mathit{txIx} & \in & \mathsf{Ix} & \text{transaction index} \\ \mathit{pp} & \in & \mathsf{PParams} & \text{protocol parameters} \\ \mathit{acnt} & \in & \mathsf{Acnt} & \text{accounting state} \end{array} \right)$$

Ledger state

$$\mathsf{LState} = \left(\begin{array}{rcll} \mathit{utxoSt} & \in & \mathsf{UTxOState} & \text{UTxO state} \\ \mathit{dpstate} & \in & \mathsf{DPState} & \text{delegation and pool state} \end{array} \right)$$

Ledger transitions

$$\_ \vdash \_ \xrightarrow[\text{LEDGER}]{} \_ \subseteq \mathbb{P}\ (\mathsf{LEnv} \times \mathsf{LState} \times \mathsf{Tx} \times \mathsf{LState})$$

Figure 30: Ledger transition-system types

Figure 30 defines the ledger state transition. It has a single rule, which first calls the UTXOW transition, then calls the DELEGS transition.