$$\text{PP-Update-Empty}\ \frac{up = \diamond}{\begin{array}{r} slot \\ pp \\ genDelegs \end{array} \vdash \begin{pmatrix} pup_s \\ fpup_s \end{pmatrix} \xrightarrow[\text{PPUP}]{up} \begin{pmatrix} pup_s \\ fpup_s \end{pmatrix}} \tag{1}$$

$$\text{PP-Update-Current}\ \frac{\begin{array}{c} (pup,\ e) := up \qquad \operatorname{dom} pup \subseteq \operatorname{dom} genDelegs \\ \forall ps \in \operatorname{range} pup,\ pv \mapsto v \in ps \implies \mathsf{pvCanFollow}\ (\mathsf{pv}\ pp)\ v \\ slot < \mathsf{firstSlot}\ ((\mathsf{epoch}\ slot) + 1) - 2 \cdot \mathsf{StabilityWindow} \\ \mathsf{epoch}\ slot = e \end{array}}{\begin{array}{r} slot \\ pp \\ genDelegs \end{array} \vdash \begin{pmatrix} pup_s \\ fpup_s \end{pmatrix} \xrightarrow[\text{PPUP}]{up} \begin{pmatrix} \boldsymbol{pup_s \underset{\rightarrow}{\cup} pup} \\ fpup_s \end{pmatrix}} \tag{2}$$

$$\text{PP-Update-Future}\ \frac{\begin{array}{c} (pup,\ e) := up \qquad \operatorname{dom} pup \subseteq \operatorname{dom} genDelegs \\ \forall ps \in \operatorname{range} pup,\ pv \mapsto v \in ps \implies \mathsf{pvCanFollow}\ (\mathsf{pv}\ pp)\ v \\ slot \geq \mathsf{firstSlot}\ ((\mathsf{epoch}\ slot) + 1) - 2 \cdot \mathsf{StabilityWindow} \\ (\mathsf{epoch}\ slot) + 1 = e \end{array}}{\begin{array}{r} slot \\ pp \\ genDelegs \end{array} \vdash \begin{pmatrix} pup_s \\ fpup_s \end{pmatrix} \xrightarrow[\text{PPUP}]{up} \begin{pmatrix} pup_s \\ \boldsymbol{fpup_s \underset{\rightarrow}{\cup} pup} \end{pmatrix}} \tag{3}$$

Figure 13: Protocol Parameter Update Inference Rules

8 UTxO

A key constraint that must always be satisfied as a result and precondition of a valid ledger state transition is called the *general accounting property*, or the *preservation of value* condition. Every piece of software that is a part of the implementation of the Cardano cryptocurrency must function in such a way as to not result in a violation of this rule. If this condition is not satisfied, it is an indicator of incorrect accounting, potentially due to malicious disruption or a bug.

The preservation of value is expressed as an equality that uses values in the ledger state and the environment, as well as the values in the body of the signal transaction. We have defined the rules of the delegation protocol in a way that should consistently satisfy the preservation of value.

In this section, we discuss the relevant accounting that needs to be done as a result of processing a transaction, i.e. the deposits for all certificates, transaction fees, transaction withdrawals and refunds for individual deregistration, so that we may keep track of whether the preservation of value is satisfied. Stake pool retirement refunds are not triggered by a transaction (but rather, happen at the epoch boundary) and are therefore not considered in our state change rules invoked due to a signal transaction.

Note that when a transaction is issued by a wallet to be applied to the ledger state (i.e. processed), the rules in this section are defined in such a way that it is impossible to apply only some parts of a transaction (e.g. only certain certificates). Every part of the transaction must be valid and it must be live, otherwise it is ignored entirely. It is the wallet's responsibility to inform the user that a transaction failed to be processed.

8.1 UTxO Transitions

Figure 14 defines functions needed for the UTxO transition system. See Figure 10 for most of the definitions used in the transition system.

- The function outs creates unspent outputs generated by a transaction, so that they can be added to the ledger state. For each output in the transaction, outs maps the transaction id and output index to the output.
- The ubalance function calculates sum total of all the coin in a given UTxO.
- The wbalance function calculates the total sum of all the reward withdrawals in a transaction.
- The calculation consumed gives the value consumed by the transaction *tx* in the context of the protocol parameters, the current UTxO on the ledger and the registered stake credentials. This calculation is a sum of all coin in the inputs of *tx*, reward withdrawals and stake credential deposit refunds. Some of the definitions used in this function will be defined in Section 8.2. In particular, keyRefunds is defined in Figure 17.
- The calculation produced gives the value produced by the transaction *tx* in the context of the protocol parameters and the registered stake pools. This calculation is a sum of all coin in the outputs of *tx*, the transaction fee and all needed deposits. Some of the definitions used in this function will be defined in Section 8.2. In particular, totalDeposits is defined in Figure 17.

For a transaction and a given ledger state, the preservation of value property holds exactly when the results of consumed equal the results of produced. Moreover, when the property holds, value is only moved between transaction outputs, the reward accounts, the fee pot and the deposit pot.

Note that the produced function takes the registered stake pools (*poolParams*) as a parameter only in order to determine which pool registration certificates are new (and thus require a deposit) and which ones are updates. Registration will be discussed more in Section 9.

$$\mathsf{outs} \in \mathsf{TxBody} \to \mathsf{UTxO} \qquad \text{tx outputs as UTxO}$$
$$\mathsf{outs}\ tx = \{(\mathsf{txid}\ tx, ix) \mapsto txout \mid ix \mapsto txout \in \mathsf{txouts}\ tx\}$$

$$\mathsf{ubalance} \in \mathsf{UTxO} \to \mathsf{Coin} \qquad \text{UTxO balance}$$
$$\mathsf{ubalance}\ utxo = \sum_{(\_ \mapsto (\_,\ c)) \in utxo} c$$

$$\mathsf{wbalance} \in \mathsf{Wdrl} \to \mathsf{Coin} \qquad \text{withdrawal balance}$$
$$\mathsf{wbalance}\ ws = \sum_{(\_ \mapsto c) \in ws} c$$

$$\mathsf{consumed} \in \mathsf{PParams} \to \mathsf{UTxO} \to \mathsf{TxBody} \to \mathsf{Coin} \qquad \text{value consumed}$$
$$\mathsf{consumed}\ pp\ utxo\ tx =$$
$$\mathsf{ubalance}\ (\mathsf{txins}\ tx \lhd utxo) + \mathsf{wbalance}\ (\mathsf{txwdrls}\ tx) + \mathsf{keyRefunds}\ pp\ tx$$

$$\mathsf{produced} \in \mathsf{PParams} \to (\mathsf{KeyHash}_{pool} \mapsto \mathsf{PoolParam}) \to \mathsf{TxBody} \to \mathsf{Coin} \qquad \text{value produced}$$
$$\mathsf{produced}\ pp\ poolParams\ tx =$$
$$\mathsf{ubalance}\ (\mathsf{outs}\ tx) + \mathsf{txfee}\ tx + \mathsf{totalDeposits}\ pp\ poolParams\ (\mathsf{txcerts}\ tx)$$

Figure 14: Functions used in UTxO rules

The types for the UTxO transition are given in Figure 15. The environment, UTxOEnv, consists of:

- The current slot.
- The protocol parameters.
- The registered stake pools (also explained in Section 9, Figure 21).
- The genesis key delegation mapping.

The current slot and registrations are needed for the refund calculations described in Section 8.2.

The state needed for the UTxO transition UTxOState, consists of:

- The current UTxO.
- The deposit pot.
- The fee pot.
- Proposed updates (see Section 7).

The signal for the UTxO transition is a transaction.

UTxO environment

$$\mathsf{UTxOEnv} = \left(\begin{array}{rcll} \mathit{slot} & \in & \mathsf{Slot} & \text{current slot} \\ \mathit{pp} & \in & \mathsf{PParams} & \text{protocol parameters} \\ \mathit{poolParams} & \in & \mathsf{KeyHash}_{\mathit{pool}} \mapsto \mathsf{PoolParam} & \text{stake pools} \\ \mathit{genDelegs} & \in & \mathsf{GenesisDelegation} & \text{genesis key delegations} \end{array}\right)$$

UTxO States

$$\mathsf{UTxOState} = \left(\begin{array}{rcll} \mathit{utxo} & \in & \mathsf{UTxO} & \mathsf{UTxO} \\ \mathit{deposited} & \in & \mathsf{Coin} & \text{deposits pot} \\ \mathit{fees} & \in & \mathsf{Coin} & \text{fee pot} \\ \mathit{ppup} & \in & \mathsf{PPUpdateState} & \text{proposed updates} \end{array}\right)$$

UTxO transitions

$$\_ \vdash \_ \xrightarrow[\mathsf{UTXO}]{} \_ \subseteq \mathbb{P}\ (\mathsf{UTxOEnv} \times \mathsf{UTxOState} \times \mathsf{Tx} \times \mathsf{UTxOState})$$

Figure 15: UTxO transition-system types

The UTxO transition system is given in Figure 16. Rule 4 specifies the conditions under which a transaction can be applied to a particular UTxOState in environment UTxOEnv:

The transition contains the following predicates:

- The transaction is live (the current slot is less than its time to live).
- The transaction has at least one input. The global uniqueness of transaction inputs prevents replay attacks. By requiring that all transactions spend at least one input, the entire transaction is safe from such attacks. A delegation certificate by itself, for example, does not have this property.

- The fee paid by the transaction has to be greater than or equal to the minimum fee, which is based on the size of the transaction. We leave open the future possibility that transactions with larger fees can be prioritized.
- Each input spent in the transaction must be in the set of unspent outputs.
- The *preservation of value* property must hold. In other words, the amount of value produced by the transaction must be the same as the amount consumed.
- The PPUP transition is successful.
- The coin value of each new output must be at least as large as the minimum value specified by the protocol parameter *minUTxOValue*.
- The transaction size must be below the allowed maximum. Note that there is an implicit max transaction size given by the max block size, and that if we wished to allow a transaction to be as large as will fit in a block, this check would not be needed. Being able to limit the size below that of the block, however, gives us some control over how transactions will be packed into the blocks.

If all the predicates are satisfied, the state is updated as follows:

- Update the UTxO:
  - Remove from the UTxO all the $(txin, txout)$ pairs associated with the *txins*'s in the *inputs* list of the transaction body *txb*.
  - Add all the *outputs* of *tx* to the UTxO, associated with the txid of the transaction body *txb*
- Add all new deposits to the deposit pot and subtract all deposit refunds.
- Add the transaction fee to the fee pot.
- Update the current update proposals.

The accounting for the reward withdrawals is not done in this transition system. The rewards are tracked with the delegation state and will be removed in the final delegation transition, see 16.

Note here that output entries for both the deposit refunds and the rewards withdrawals must be included in the body of the transaction carrying the deregistration certificates (requesting these refunds) and the reward requests. It is the job of the wallet to calculate the value of these refunds and withdrawals and generate the correct outputs to include in the outputs list of *tx* such that applying this transaction results in a valid ledger update adding correct amounts of coin to the right addresses.

The approach of including refunds and rewards directly in the $outputs$ gives great flexibility to the management of the coin value obtained from these accounts, i.e. it can be directed to any address. However, it means there is no direct link between the *wdrls* requests (similarly for the key deregistration certificate addresses and refund amounts) and the $outputs$. We verify that the included outputs are correct and authorized through the preservation of value condition and witnessing the transaction. The combination of the preservation of value and witnessing, described in Section 8.3, assures that the ledger state is updated correctly.

The main difference, however, in how rewards and refunds work is that refunds come from the *deposited* pot, which is a single coin value indicating the sum of all the deposits, while rewards come from individual accounts where a reward is accumulated to a specific address.

The UTXO rule has ten predicate failures: