$$\text{ledger}\ \frac{\begin{array}{c}\begin{array}{c}\mathit{slot}\\ \mathit{txIx}\\ \mathit{pp}\\ \mathit{tx}\\ \mathit{acnt}\end{array} \vdash \mathit{dpstate} \xrightarrow[\mathsf{DELEGS}]{\mathsf{txcerts}\ (\mathsf{txbody}\ \mathit{tx})} \mathit{dpstate}' \\[1em] (\mathit{dstate}, \mathit{pstate}) := \mathit{dpstate} \\ (\_,\_,\_,\_,\mathit{genDelegs},\_) := \mathit{dstate} \\ (\mathit{poolParams},\_,\_) := \mathit{pstate} \\[1em] \begin{array}{c}\mathit{slot}\\ \mathit{pp}\\ \mathit{poolParams}\\ \mathit{genDelegs}\end{array} \vdash \mathit{utxoSt} \xrightarrow[\mathsf{UTXOW}]{\mathit{tx}} \mathit{utxoSt}'\end{array}}{\begin{array}{c}\mathit{slot}\\ \mathit{txIx}\\ \mathit{pp}\\ \mathit{acnt}\end{array} \vdash \begin{pmatrix}\mathit{utxoSt}\\ \mathit{dpstate}\end{pmatrix} \xrightarrow[\mathsf{LEDGER}]{\mathit{tx}} \begin{pmatrix}\boldsymbol{\mathit{utxoSt}'}\\ \boldsymbol{\mathit{dpstate}'}\end{pmatrix}} \tag{18}$$

Figure 31: Ledger inference rule

The transition system LEDGER in Figure 31 is iterated in LEDGERS in order to process a list of transactions.

Ledger Sequence transitions

$$\_ \vdash \_ \xrightarrow[\text{LEDGERS}]{\_} \_ \subseteq \mathbb{P}\left((\mathsf{Slot} \times \mathsf{PParams} \times \mathsf{Coin}) \times \mathsf{LState} \times \mathsf{Tx}^{*} \times \mathsf{LState}\right)$$

Figure 32: Ledger Sequence transition-system types

$$\text{Seq-ledger-base}\ \frac{}{\begin{array}{r} \mathit{slot} \\ \mathit{pp} \\ \mathit{acnt} \end{array} \vdash \mathit{ls} \xrightarrow[\text{LEDGERS}]{\epsilon} \mathit{ls}} \tag{19}$$

$$\text{Seq-ledger-ind}\ \frac{\begin{array}{r} \mathit{slot} \\ \mathit{pp} \\ \mathit{acnt} \end{array} \vdash \mathit{ls} \xrightarrow[\text{LEDGERS}]{\Gamma} \mathit{ls}' \qquad \begin{array}{r} \mathit{slot} \\ \mathsf{len}\ \Gamma \\ \mathit{pp} \\ \mathit{acnt} \end{array} \vdash \mathit{ls}' \xrightarrow[\text{LEDGER}]{\mathit{tx}} \mathit{ls}''}{\begin{array}{r} \mathit{slot} \\ \mathit{pp} \\ \mathit{acnt} \end{array} \vdash \mathit{ls} \xrightarrow[\text{LEDGERS}]{\Gamma;\ \mathit{tx}} \mathit{ls}''} \tag{20}$$

Figure 33: Ledger sequence rules

11 Rewards and the Epoch Boundary

This chapter introduces the epoch boundary transition system and the related reward calculation.

The transition system is defined in Section 11.8, and involves taking stake distribution snapshots (Sections 11.4 and 11.5), retiring stake pools (Section 11.6), and performing protocol updates (Section 11.7). The reward calculation, defined in Sections 11.9 and 11.10, distributes the leader election rewards.

11.1 Overview of the Reward Calculation

The rewards for a given epoch e_i involve the two epochs surrounding it. In particular, the stake distribution will come from the previous epoch and the rewards will be calculated in the following epoch. More concretely:

(A) A stake distribution snapshot is taken at the begining of epoch e_{i-1}.

(B) The randomness for leader election is fixed during epoch e_{i-1}

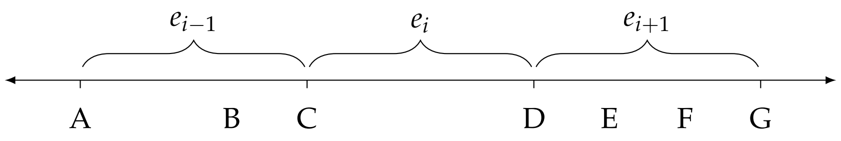
(C) Epoch e_i begins.

(D) Epoch e_i ends. A snapshot is taken of the stake pool performance during epoch e_i. A snapshot is also taken of the fee pot.

(E) The snapshots from (D) are stable and the reward calculation can begin.

(F) The reward calculation is finished and an update to the ledger state is ready to be applied.

(G) Rewards are given out.



We must therefore store the last three stake distributions. The mnemonic "mark, set, go" will be used to keep track of the snapshots, where the label "mark" refers to the most recent snapshot, and "go" refers to the snapshot that is ready to be used in the reward calculation. In the above diagram, the snapshot taken at (A) is labeled "mark" during epoch e_{i-1}, "set" during epoch e_i and "go" during epoch e_{i+1}. At (G) the snapshot taken at (A) is no longer needed and will be discarded.

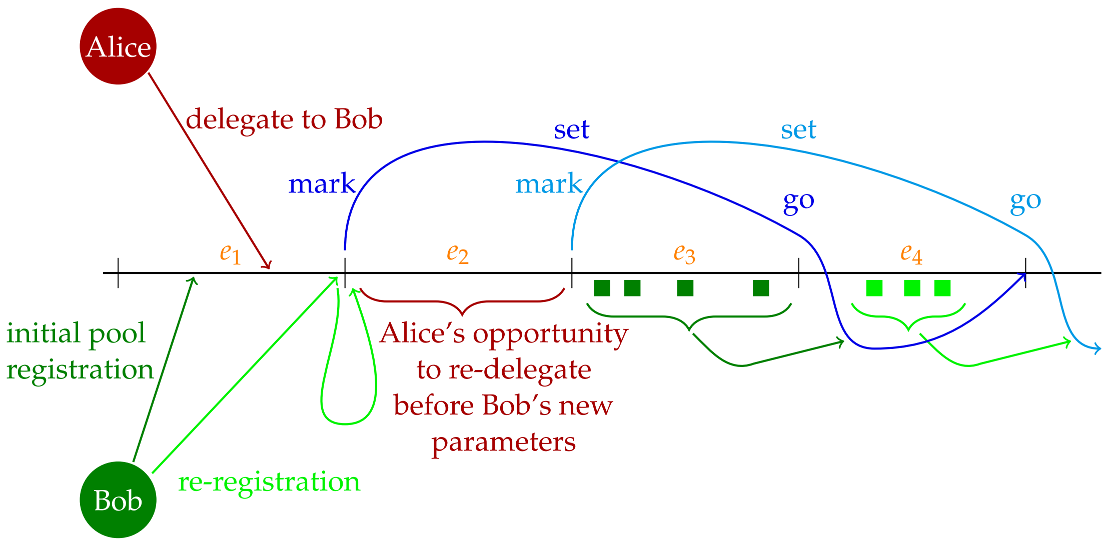
The two main transition systems in this section are:

- The transition system named EPOCH, which is defined in Section 11.8, covers what happens at the epoch boundary, such as at (A), (C), (D) and (G).
- The transition named RUPD, which is defined in Section 12.6, covers the reward calculation that happens between (E) and (F).

NOTE

Between time D and E we are concerned with chain growth and stability. Therefore this duration can be stated as 2k blocks (to state it in slots requires details about the particular version of the Ouroboros protocol). The duration between F and G is also 2k blocks. Between E and F a single honest block is enough to ensure a random nonce.

11.2 Example Illustration of the Reward Cycle


Alice
delegate to Bob
set
set
mark
mark
go
go
e_1
e_2
e_3
e_4
initial pool registration
Alice's opportunity to re-delegate before Bob's new parameters
Bob
re-registration

Bob registers his stake pool in epoch e_1. Alice delegates to Bob's stake pool in epoch e_1. Just before the end of epoch e_1, Bob submits a stake pool re-registration, changing his pool parameters. The change in parameters is not immediate, as shown by the curved arrow around the epoch boundary.

A snapshot is taken on the e_1/e_2 boundary. It is labeled "mark" initially. This snapshot includes Alice's delegation to Bob's pool, and Bob's pool parameters and listed in the initial pool registration certificate.

If Alice changes her delegation choice any time during epoch e_2, she will never be effected by Bob's change of parameters.

A new snapshot is taken on the e_2/e_3 boundary. The previous (darker blue) snapshot is now labeled "set", and the new one labeled "mark". The "set" snapshot is used for leader election in epoch e_3.

On the e_3/e_4 boundary, the darker blue snapshot is labeled "go" and the lighter blue snapshot is labeled "set". Bob's stake pool performance during epoch e_3 (he produced 4 blocks) will be used with the darker blue snapshot for the rewards which will be handed out at the beginning of epoch e_5.

11.3 Helper Functions and Accounting Fields

Figure 34 defines four helper functions needed throughout the rest of the section.

- The function obligation calculates the the minimal amount of coin needed to pay out all deposit refunds.
- The function poolStake filters the stake distribution to one stake pool.

The Figure 35 lists the accounting fields, denoted by Acnt, which will be used throughout this section. It consists of:

- The value *treasury* tracks the amount of coin currently stored in the treasury. Initially there will be no way to remove these funds.
- The value *reserves* tracks the amount of coin currently stored in the reserves. This pot is used to pay rewards.

More will be said about the general accounting system in Section 11.10.

Total possible refunds

$$\mathsf{obligation} \in \mathsf{PParams} \to (\mathsf{Credential}_{stake} \mapsto \mathsf{Coin}) \to (\mathsf{KeyHash}_{pool} \mapsto \mathsf{PoolParam}) \to \mathsf{Coin}$$

$$\begin{aligned}&\mathsf{obligation}\ pp\ rewards\ poolParams = \\ &\quad (\mathsf{keyDeposit}\ pp) \cdot |rewards| + (\mathsf{poolDeposit}\ pp) \cdot |poolParams|\end{aligned}$$

Filter Stake to one Pool

$$\begin{aligned}&\mathsf{poolStake} \in \mathsf{KeyHash}_{pool} \to (\mathsf{KeyHash}_{stake} \mapsto \mathsf{KeyHash}_{pool}) \to \mathsf{Stake} \to \mathsf{Stake} \\ &\mathsf{poolStake}\ hk\ delegs\ stake = \mathrm{dom}\,(delegs \rhd \{hk\}) \lhd stake\end{aligned}$$

Figure 34: Helper Functions used in Rewards and Epoch Boundary

Accounting Fields

$$\mathsf{Acnt} = \left(\begin{array}{rcll} treasury & \in & \mathsf{Coin} & \text{treasury pot} \\ reserves & \in & \mathsf{Coin} & \text{reserve pot}\end{array}\right)$$

Figure 35: Accounting fields

11.4 Stake Distribution Calculation

This section defines the stake distribution calculations. Figure 36 introduces three new derived types:

- BlocksMade represents the number of blocks each stake pool produced during an epoch.
- Stake represents the amount of stake (in Coin) controlled by each stake pool.

Derived types

$$\begin{array}{rclcll} blocks & \in & \mathsf{BlocksMade} & = & \mathsf{KeyHash}_{pool} \mapsto \mathbb{N} & \text{blocks made by stake pools} \\ stake & \in & \mathsf{Stake} & = & \mathsf{Credential} \mapsto \mathsf{Coin} & \text{stake}\end{array}$$

Figure 36: Epoch definitions

The stake distribution calculation is given in Figure 37.

- $\mathsf{aggregate}_{+}$ takes a relation on $A \times B$, where B is any monoid $(B, +, e)$ and returns a map from each $a \in A$ to the "sum" (using the monoidal $+$ operation) of all $b \in B$ such that $(a, b) \in A \times B$.
- $\mathsf{stakeDistr}$ uses the $\mathsf{aggregate}_{+}$ function and several relations to compute the stake distribution, mapping each hashkey to the total coin under its control. Keys that are not both registered and delegated are filtered out. The relation passed to $\mathsf{aggregate}_{+}$ is made up of:
  - $\mathsf{stakeCred_b}^{-1}$, relating credentials to (base) addresses
  - $(\mathsf{addrPtr} \circ ptr)^{-1}$, relating credentials to (pointer) addresses
  - range $utxo$, relating addresses to coins
  - $\mathsf{stakeCred_r}^{-1} \circ rewards$, relating (reward) addresses to coins

  The notation for relations is explained in Section 2.