Abstract types

$$
\begin{array}{rcll}
\mathit{slot} & \in & \mathsf{Slot} & \text{absolute slot} \\
\mathit{ix} & \in & \mathsf{Ix} & \text{index} \\
\mathit{net} & \in & \mathsf{Network} & \text{either Testnet or Mainnet}
\end{array}
$$

Derived types

$$
\begin{array}{rcll}
\mathit{cred} \in \mathsf{Credential} & = & \mathsf{KeyHash} \uplus \mathsf{ScriptHash} & \\
(s,t,c) \in \mathsf{Ptr} & = & \mathsf{Slot} \times \mathsf{Ix} \times \mathsf{Ix} & \text{certificate pointer} \\
\mathit{addr} \in \mathsf{Addr_{base}} & = & \mathsf{Network} \times \mathsf{Credential}_{\mathit{pay}} \times \mathsf{Credential}_{\mathit{stake}} & \text{base address} \\
\mathit{addr} \in \mathsf{Addr_{ptr}} & = & \mathsf{Network} \times \mathsf{Credential}_{\mathit{pay}} \times \mathsf{Ptr} & \text{pointer address} \\
\mathit{addr} \in \mathsf{Addr_{enterprise}} & = & \mathsf{Network} \times \mathsf{Credential}_{\mathit{pay}} & \text{enterprise address} \\
\mathit{addr} \in \mathsf{Addr_{bootstrap}} & = & \mathsf{Network} \times \mathsf{KeyHash}_{\mathit{pay}} & \text{bootstrap address} \\
\mathit{addr} \in \mathsf{Addr} & = & \begin{array}{c}\mathsf{Addr_{base}} \uplus \mathsf{Addr_{ptr}} \uplus \mathsf{Addr_{enterprise}} \\ \uplus \mathsf{Addr_{bootstrap}}\end{array} & \text{output address} \\
\mathit{acct} \in \mathsf{Addr_{rwd}} & = & \mathsf{Network} \times \mathsf{Credential}_{\mathit{stake}} & \text{reward account}
\end{array}
$$

Address subtypes

$$
\begin{array}{rcl}
\mathit{addr}^{\mathit{vkey}}_{\mathit{base}} \in \mathsf{Addr}^{\mathit{vkey}}_{\mathit{base}} & = & \mathsf{KeyHash} \lhd \mathsf{Addr}_{\mathit{base}} \\
\mathit{addr}^{\mathit{script}}_{\mathit{base}} \in \mathsf{Addr}^{\mathit{vkey}}_{\mathit{base}} & = & \mathsf{ScriptHash} \lhd \mathsf{Addr}_{\mathit{base}} \\
\mathit{addr}^{\mathit{vkey}}_{\mathit{ptr}} \in \mathsf{Addr}^{\mathit{vkey}}_{\mathit{ptr}} & = & \mathsf{KeyHash} \lhd \mathsf{Addr}_{\mathit{ptr}} \\
\mathit{addr}^{\mathit{script}}_{\mathit{ptr}} \in \mathsf{Addr}^{\mathit{vkey}}_{\mathit{ptr}} & = & \mathsf{ScriptHash} \lhd \mathsf{Addr}_{\mathit{ptr}} \\
\mathit{addr}^{\mathit{vkey}}_{\mathit{enterprise}} \in \mathsf{Addr}^{\mathit{vkey}}_{\mathit{enterprise}} & = & \mathsf{Addr}_{\mathit{enterprise}} \cap \mathsf{KeyHash} \\
\mathit{addr}^{\mathit{script}}_{\mathit{enterprise}} \in \mathsf{Addr}^{\mathit{vkey}}_{\mathit{enterprise}} & = & \mathsf{Addr}_{\mathit{enterprise}} \cap \mathsf{ScriptHash} \\
 & & \\
\mathit{addr}^{\mathit{vkey}} \in \mathsf{Addr}^{\mathit{vkey}} & = & \mathsf{Addr}^{\mathit{vkey}}_{\mathit{base}} \uplus \mathsf{Addr}^{\mathit{vkey}}_{\mathit{ptr}} \uplus \mathsf{Addr}^{\mathit{vkey}}_{\mathit{enterprise}} \uplus \mathsf{Addr}_{\mathit{bootstrap}} \\
 & & \\
\mathit{addr}^{\mathit{script}} \in \mathsf{Addr}^{\mathit{script}} & = & \mathsf{Addr}^{\mathit{script}}_{\mathit{base}} \uplus \mathsf{Addr}^{\mathit{script}}_{\mathit{ptr}} \uplus \mathsf{Addr}^{\mathit{script}}_{\mathit{enterprise}} \\
 & & \\
\mathit{addr}^{\mathit{vkey}}_{\mathit{rwd}} \in \mathsf{Addr}^{\mathit{vkey}}_{\mathit{rwd}} & = & \mathsf{Addr}_{\mathit{rwd}} \cap \mathsf{KeyHash} \\
\mathit{addr}^{\mathit{script}}_{\mathit{rwd}} \in \mathsf{Addr}^{\mathit{script}}_{\mathit{rwd}} & = & \mathsf{Addr}_{\mathit{rwd}} \cap \mathsf{ScriptHash}
\end{array}
$$

Accessor Functions

$$
\begin{array}{rcll}
\mathsf{paymentHK} & \in & \mathsf{Addr^{vkey}} \to \mathsf{KeyHash}_{\mathit{pay}} & \text{hash of payment key from addr} \\
\mathsf{validatorHash} & \in & \mathsf{Addr^{script}} \to \mathsf{ScriptHash} & \text{hash of validator script} \\
\mathsf{stakeCred_b} & \in & \mathsf{Addr_{base}} \to \mathsf{Credential}_{\mathit{stake}} & \text{stake credential from base addr} \\
\mathsf{stakeCred_r} & \in & \mathsf{Addr_{rwd}} \to \mathsf{Credential}_{\mathit{stake}} & \text{stake credential from reward addr} \\
\mathsf{addrPtr} & \in & \mathsf{Addr_{ptr}} \to \mathsf{Ptr} & \text{pointer from pointer addr} \\
\mathsf{netId} & \in & \mathsf{Addr} \to \mathsf{Network} & \text{network Id from addr}
\end{array}
$$

Constructor Functions

$$
\begin{array}{rcll}
\mathsf{addr_{rwd}} & \in & \mathsf{Credential}_{\mathit{stake}} \to \mathsf{Addr_{rwd}} & \begin{array}{l}\text{construct a reward account,} \\ \text{implicitly using } \mathsf{NetworkId}\end{array}
\end{array}
$$

Constraints

$$
\mathit{hk_1} = \mathit{hk_2} \iff \mathsf{addr_{rwd}}\ \mathit{hk_1} = \mathsf{addr_{rwd}}\ \mathit{hk_2} \quad (\mathsf{addr_{rwd}} \text{ is injective})
$$

Figure 6: Definitions used in Addresses

5 Protocol Parameters

5.1 Updatable Protocol Parameters

The Shelley protocol parameters are listed in Figure 7. Some of the Shelley protocol parameters are common to the Byron era, specifically, the common ones are a, b, $maxTxSize$, and $maxHeaderSize$ (see the document [BL-D1]).

The type Ppm represents the names of the protocol parameters, and $\mathsf{T}_{\mathsf{ppm}}$ is the type of the protocol parameter ppm. The type PParams is a finite map containing all the Shelley parameters, indexed by their names. We will explain the significance of each parameter as it comes up in the calculations used in transition rules. The type PParamsUpdate is similar to PParams, but is a partial mapping of the protocol parameters. It is used in the update system explained in Section 7.

The type Coin is defined as an alias for the integers. Negative values will not be allowed in UTxO outputs or reward accounts, and $\mathbb{Z}$ is only chosen over $\mathbb{N}$ for its additive inverses.

Some helper functions are defined in Figure 9. The minfee function calculates the minimum fee that must be paid by a transaction. This value depends on the protocol parameters and the size of the transaction.

Two time related types are introduced, Epoch and Duration. A Duration is the difference between two slots, as given by $\_ -_s \_$.

Lastly, there are two functions, epoch and firstSlot for converting between epochs and slots and one function kesPeriod for getting the cycle of a slot. Note that Slot is an abstract type, while the constants are integers. We use multiplication and division symbols on these distinct types without being explicit about the types and conversion.

5.2 Global Constants

In additon to the updatable protocol parameters defined in Section 5.1, there are ten parameters which cannot be changed by the update system in Section 7. We call these the global constants, as changing these values can only be done by updating the software, i.e. a soft or a hard fork. For the software update mechanism, see Section 13.

The constants SlotsPerEpoch and SlotsPerKESPeriod represent the number of slots in an epoch/KES period (for a brief explanation of a KES cryptography, see Section 3). The KES period is defined as a duration of slots which determines how often the KES private key must be evolved. The KES private key must be evolved once at the start of each new KES period. The duration of the KES period is determined by SlotsPerKESPeriod. Any given KES key can be evolved up to MaxKESEvo-many times before a new operational certificate must be issued. The constants StabilityWindow and RandomnessStabilisationWindow concern the chain stability. The maximum number of time a KES key can be evolved before a pool operator must create a new operational certificate is given by MaxKESEvo. **Note that if MaxKESEvo is changed, the KES signature format may have to change as well.**

The constant Quorum determines the quorum amount needed for votes on the protocol parameter updates and the application version updates.

The constant MaxMajorPV provides a mechanism for halting outdated nodes. Once the major component of the protocol version in the protocol parameters exceeds this value, every subsequent block is invalid. See Figures 74 and 75.

The constant MaxLovelaceSupply gives the total number of lovelace in the system, which is used in the reward calculation. It is always equal to the sum of the values in the UTxO, plus the sum of the values in the reward accounts, plus the deposit pot, plus the fee pot, plus the treasury and the reserves.

The constant ActiveSlotCoeff is the value f from the Praos paper [DGKR17].

Lastly, NetworkId determines what network, either mainnet or testnet, is expected. This value will also appear inside every address, and transactions containing addresses with an unexpected network ID are rejected.

Abstract types

$p \in \mathsf{Ppm}$	protocol parameter
$dur \in \mathsf{Duration}$	difference between slots
$epoch \in \mathsf{Epoch}$	epoch
$kesPeriod \in \mathsf{KESPeriod}$	KES period

Derived types

$pp \in \mathsf{PParams}$	$= \mathsf{Ppm} \to \mathsf{T_{ppm}}$	protocol parameters
$ppup \in \mathsf{PParamsUpdate}$	$= \mathsf{Ppm} \mapsto \mathsf{T_{ppm}}$	protocol parameter update
$coin \in \mathsf{Coin}$	$= \mathbb{Z}$	unit of value
$pv \in \mathsf{ProtVer}$	$= \mathbb{N} \times \mathbb{N}$	protocol version

Protocol Parameters

$a \mapsto \mathbb{Z} \in \mathsf{PParams}$	min fee factor
$b \mapsto \mathbb{Z} \in \mathsf{PParams}$	min fee constant
$maxBlockSize \mapsto \mathbb{N} \in \mathsf{PParams}$	max block body size
$maxTxSize \mapsto \mathbb{N} \in \mathsf{PParams}$	max transaction size
$maxHeaderSize \mapsto \mathbb{N} \in \mathsf{PParams}$	max block header size
$poolDeposit \mapsto \mathsf{Coin} \in \mathsf{PParams}$	stake pool deposit
$E_{max} \mapsto \mathsf{Epoch} \in \mathsf{PParams}$	epoch bound on pool retirement
$n_{opt} \mapsto \mathbb{N}^{>0} \in \mathsf{PParams}$	desired number of pools
$a_0 \mapsto [0, \infty) \in \mathsf{PParams}$	pool influence
$\tau \mapsto [0, 1] \in \mathsf{PParams}$	treasury expansion
$\rho \mapsto [0, 1] \in \mathsf{PParams}$	monetary expansion
$d \mapsto \{0, 1/100, 2/100, \ldots, 1\} \in \mathsf{PParams}$	decentralization parameter
$extraEntropy \mapsto \mathsf{Seed} \in \mathsf{PParams}$	extra entropy
$pv \mapsto \mathsf{ProtVer} \in \mathsf{PParams}$	protocol version
$minUTxOValue \mapsto \mathsf{Coin} \in \mathsf{PParams}$	minimum allowed value of a new TxOut
$minPoolCost \mapsto \mathsf{Coin} \in \mathsf{PParams}$	minimum allowed stake pool cost

Accessor Functions

a, b, maxBlockSize, maxTxSize, maxHeaderSize, keyDeposit, poolDeposit, emax, nopt, influence, tau, rho, d, extraEntropy, pv, minUTxOValue, minPoolCost

Abstract Functions

$(\ -_s\ ) \in \mathsf{Slot} \to \mathsf{Slot} \to \mathsf{Duration}$	duration between slots

Figure 7: Definitions Used in Protocol Parameters

Global Constants

$\mathsf{SlotsPerEpoch} \in \mathbb{N}$	- slots per epoch
$\mathsf{SlotsPerKESPeriod} \in \mathbb{N}$	- slots per KES period
$\mathsf{StabilityWindow} \in \mathsf{Duration}$	- window size for chain growth guarantees, see in [DGKR17]
$\mathsf{RandomnessStabilisationWindow} \in \mathsf{Duration}$	- duration needed for epoch nonce stabilization
$\mathsf{MaxKESEvo} \in \mathbb{N}$	- maximum KES key evolutions
$\mathsf{Quorum} \in \mathbb{N}$	- quorum for update system votes
$\mathsf{MaxMajorPV} \in \mathbb{N}$	- all blocks are invalid after this value
$\mathsf{MaxLovelaceSupply} \in \mathsf{Coin}$	- total lovelace in the system
$\mathsf{ActiveSlotCoeff} \in (0,1]$	- f in [DGKR17]
$\mathsf{NetworkId} \in \mathsf{Network}$	- the network, mainnet or testnet

Figure 8: Global Constants

Helper Functions

$$\mathsf{minfee} \in \mathsf{PParams} \to \mathsf{Tx} \to \mathsf{Coin}$$
$$\mathsf{minfee}\ pp\ tx = (\mathsf{a}\ pp) \cdot \mathsf{txSize}\ tx + (\mathsf{b}\ pp)$$

minimum fee

$$\mathsf{epoch} \in \mathsf{Slot} \to \mathsf{Epoch}$$
$$\mathsf{epoch}\ slot = slot\ /\ \mathsf{SlotsPerEpoch}$$

epoch of a slot

$$\mathsf{firstSlot} \in \mathsf{Epoch} \to \mathsf{Slot}$$
$$\mathsf{firstSlot}\ e = e \cdot \mathsf{SlotsPerEpoch}$$

first slot of an epoch

$$\mathsf{kesPeriod} \in \mathsf{Slot} \to \mathsf{KESPeriod}$$
$$\mathsf{kesPeriod}\ slot = slot\ /\ \mathsf{SlotsPerKESPeriod}$$

KES period of a slot

Figure 9: Helper functions for the Protocol Parameters