Blockchain System

1 Definitions

Definition 1 (Set). *Let* **Elt** *be the set of (concrete) elements. Let* $\emptyset$ *be an empty set and* $\mathbf{e} \in \mathbf{Elt}$. *A set of elements is expressed as the following syntax:*
$\mathbf{s} ::= \emptyset \mid \mathbf{e} \mid \mathbf{s} :: \mathbf{s}$

Definition 2 (Account). *An account is expressed as a tuple* $<$ **pak**, **puk** $>$, *where* **pak** *is its private key and* **puk** *is its public key.*

Definition 3 (Manager). *An manager manages an account on the blockchian. It is expressed as a tuple* $<$ **puk**, **bal**, **cou** $>$, *where* **puk** *is the public key of an account,* **bal** *is its balance and* **cou** *is the counter for its operations that could be T or F.*

Let **K** be a set of managers. The following functions interact with **K**.

```
let rec checkBal K puk n m =
  match K with
  | 0 -> true
  | < puk’, bal, cou > :: K’ ->
    if (puk = puk’) and (n + m) <= bal then true
    else checkBal (K’, puk, n, m)
```

```
let rec checkExi K puk =
  match K with
  | 0 -> false
  | < puk’, bal, cou > :: K’ ->
    if (puk = puk’) then true
    else checkExi (K’, puk)
```

```
let rec checkCou K puk =
  match K with
  | 0 -> false
  | < puk’, bal, cou > :: K’ ->
    if (puk = puk’) and (cou = T) then true
    else checkCou (K’, puk)
```

```
let rec updateCou K puk =
  match K with
  | 0 -> 0
  | < puk’, bal, cou > :: K’ ->
    if (puk = puk’) then < puk’, bal, F > :: K’
    else < puk’, bal, cou > :: updateCou (K’, puk)
```

```
let rec updateSuc K puk puk' m n =
  match K with
  | 0 -> 0
  | < puk'', bal, cou > :: K' ->
    if (puk = puk'') then < puk'', bal - (n + m), T >
       :: updateSuc (K', puk, puk', n, m)
    else if (puk' = puk'') then < puk'', bal + n, cou > :: K'
         else  < puk'', bal, cou >
               ::updateSuc (K', puk, puk', n, m)
```

Definition 4 (Operation). *An operation is expressed as the following syntax:* **op** *::= transfer* **n** *from* **puk** *to* **puk′** *fee* **m**

Let **C** be the set of accounts and **O** be a set of operations.

Definition 5 (State of a node). *The state of a node is expressed as a pair* [**C**, **O**] *.*

When an operation is injected in a node, it enters in a pending pool (and called a pending move).

Definition 6 (Pending operation). *A pending operation is expressed as a pair* $<$ **op**, **t** $>$*, where* **op** *is an operation and* **t** *is the time when it is injected.*

After sometime, a pending operation could be included in the blockchain as a accepted operation.

Definition 7 (Accepted operation). *An accepted operation is expressed as a tuple* $<$ **op**, **t** $>$*, where* **h** *is the hash of the block head that includes the operation.*

Let **P** be a set of pending operations, **A** be a set of accepted operations, and **t** is the current time of the blockchain.

Definition 8 (Blockchain). *The state of a blockchain is expressed as a tuple* [**P**, **A**, **K**, **t**].

Definition 9 (Blockchain system). *A blockchain system* $\mathbf{S} \triangleq \langle\ \mathbf{M}, \mathbf{B}\ \rangle$ *consists of*

1. $\mathbf{M} \equiv$ *[C, O] is the state of a node, and*
2. $\mathbf{B} \equiv$ *[P, A, K, t] is the state of a blockchain such as* $\forall\ c \in C \implies \exists\ k \in K$, *k.puk = c.puk.*

2 Rules

Rule 1 [injected]:

$$\frac{\text{checkBan(K, } puk, m, n) \wedge \text{checkCou(K, } puk) \wedge \text{checkExi(K, } puk')}{\begin{array}{c}\langle\ [< pak, puk > :: \text{C, O}], \text{[P, A, K, t]}\ \rangle \to \langle\ \text{[C, O]}, [(< \text{transfer } n \\ \text{from } puk \text{ to } puk' \text{ fee } m, \text{t} >) :: \text{P, A, updateCou(K, puk), t]}\ \rangle\end{array}} \tag{1}$$

Rule 2 [included]:

$$\frac{}{\begin{array}{c}\text{[< transfer } n \text{ from } puk \text{ to } puk' \text{ fee } m\text{, t > :: P, A, K, t']} \rightarrow \text{[P,}\\ \text{< transfer n from puk to puk' fee m, t' > :: A, updateSuc(K, puk,}\\ \text{puk', n, m), t' + 1]}\end{array}} \tag{2}$$

Rule 3 [timeout]:

$$\frac{\text{t' - t >= 60}}{\begin{array}{c}\text{[< transfer } n \text{ from } puk \text{ to } puk' \text{ fee } m\text{, t > :: P, A, K, t']} \rightarrow\\ \text{[P, A, updateFai(K, puk), t']}\end{array}} \tag{3}$$