MODULE *SplitLess_replica_group_expenses*

EXTENDS *Naturals*, *Sequences*, *FiniteSets*

CONSTANTS
USERS,
POSSIBLE_SHARES,
POSSIBLE_EXPENSE_IDs,
POSSIBLE_GROUP_IDs,
NO_EXPENSE,
NO_GROUP,
POSSIBLE_REPLICA_IDs,
ASSIGNED_REPLICA

VARIABLES *replicas*, *actionCounter*

Records

$Expense \triangleq$
$[id : POSSIBLE\_EXPENSE\_IDs,$
$group : POSSIBLE\_GROUP\_IDs \cup \{NO\_GROUP\},$ current group for easier access
$groupCounter : [POSSIBLE\_GROUP\_IDs \to Nat],$ causal length counter for group membership
$version : Nat,$ current version of expense, only payer can edit expense and each user only works on at most one re
$payer : USERS,$
$amount : Nat,$
$shares : POSSIBLE\_SHARES,$
$deleted :$ BOOLEAN $]$

$Group \triangleq$
$[id : POSSIBLE\_GROUP\_IDs,$
$creator : USERS,$
$members : [USERS \to Nat],$ Casul length counter for each user
$expenseids :$ SUBSET $POSSIBLE\_EXPENSE\_IDs,$
$totalGifted : Nat,$
$individualGiftsSent : [USERS \to Nat]]$

$Replica \triangleq$
$[id : POSSIBLE\_REPLICA\_IDs,$
$recordedExpenses : [POSSIBLE\_EXPENSE\_IDs \to (Expense \cup \{NO\_EXPENSE\})],$
$groups : [POSSIBLE\_GROUP\_IDs \to (Group \cup \{NO\_GROUP\})]$
$]$

Initialization

$Init \triangleq$

$\land$ *replicas* =
  [*rid* $\in$ *POSSIBLE_REPLICA_IDs* $\mapsto$
    [*id* $\mapsto$ *rid*,
     recordedExpenses $\mapsto$ [*eid* $\in$ *POSSIBLE_EXPENSE_IDs* $\mapsto$ *NO_EXPENSE*],
     groups $\mapsto$ [*gid* $\in$ *POSSIBLE_GROUP_IDs* $\mapsto$ *NO_GROUP*]
    ]
  ]
$\land$ *actionCounter* = 0

Helper Functions

RECURSIVE *SumFunction*(_)
SumFunction(*F*) $\triangleq$
  IF DOMAIN *F* = {} THEN 0
  ELSE LET *d* $\triangleq$ CHOOSE *x* $\in$ DOMAIN *F* : TRUE
       IN *F*[*d*] + *SumFunction*([*y* $\in$ DOMAIN *F* $\setminus$ {*d*} $\mapsto$ *F*[*y*]])

Balance(*u*, *gid*, *replica*) $\triangleq$
  LET *groupExpenses* $\triangleq$
        {*eid* $\in$ *replica.groups*[*gid*].*expenseids* :
          $\land$ *replica.recordedExpenses*[*eid*] $\neq$ *NO_EXPENSE*
          $\land$ *replica.recordedExpenses*[*eid*].*deleted* = FALSE}
  IN *SumFunction*([*eid* $\in$ *groupExpenses* $\mapsto$
                    IF *replica.recordedExpenses*[*eid*].*payer* = *u*
                    THEN *replica.recordedExpenses*[*eid*].*amount* ELSE 0])
     $-$ *SumFunction*([*eid* $\in$ *groupExpenses* $\mapsto$ *replica.recordedExpenses*[*eid*].*shares*[*u*]])
     $-$ *replica.groups*[*gid*].*individualGiftsSent*[*u*]

ComputeBalances(*grp*, *recordedExpensesIn*) $\triangleq$
  [*u* $\in$ *USERS* $\mapsto$
    LET *groupExpenses* $\triangleq$
          {*eid* $\in$ *grp.expenseids* :
            recordedExpensesIn[*eid*] $\neq$ *NO_EXPENSE* $\land$ *recordedExpensesIn*[*eid*].*deleted* = FALSE}
    IN *SumFunction*([*eid* $\in$ *groupExpenses* $\mapsto$
                      IF *recordedExpensesIn*[*eid*].*payer* = *u*
                      THEN *recordedExpensesIn*[*eid*].*amount* ELSE 0])
       $-$ *SumFunction*([*eid* $\in$ *groupExpenses* $\mapsto$ *recordedExpensesIn*[*eid*].*shares*[*u*]])
  ]

ComputeGifts(*grp*, *balances*) $\triangleq$
  LET *giftingUsers* $\triangleq$
        {*u* $\in$ *USERS* : *grp.members*[*u*]%2 = 0 $\land$ *balances*[*u*] > 0}

$$
\begin{array}{l}
\quad\quad newTotalGifted \triangleq SumFunction([u \in giftingUsers \mapsto balances[u]]) \\
\quad\quad newIndividualGifts \triangleq [u \in USERS \mapsto \text{IF } u \in giftingUsers \text{ THEN } balances[u] \text{ ELSE } 0] \\
\text{IN} \quad [grp \text{ EXCEPT } !.totalGifted = newTotalGifted, \\
\quad\quad\quad\quad\quad\quad !.individualGiftsSent = newIndividualGifts]
\end{array}
$$

$$
\begin{array}{l}
RecalcGifts(groupsIn,\ recordedExpensesIn) \triangleq \\
\quad [gid \in POSSIBLE\_GROUP\_IDs \mapsto \\
\quad\quad \text{IF } groupsIn[gid] = NO\_GROUP \text{ THEN } NO\_GROUP \\
\quad\quad \text{ELSE LET } grp \triangleq groupsIn[gid] \\
\quad\quad\quad\quad\quad balances \triangleq ComputeBalances(grp,\ recordedExpensesIn) \\
\quad\quad\quad\quad \text{IN} \quad ComputeGifts(grp,\ balances) \\
\quad]
\end{array}
$$

Group actions

$$
\begin{array}{l}
CreateGroup \triangleq \\
\quad \exists\, actor \in USERS : \\
\quad \exists\, gid \in POSSIBLE\_GROUP\_IDs : \\
\quad \exists\, rid \in POSSIBLE\_REPLICA\_IDs : \\
\quad\quad \land\ ASSIGNED\_REPLICA[actor] = rid \\
\quad\quad \land\ \forall\, otherRid \in POSSIBLE\_REPLICA\_IDs : replicas[otherRid].groups[gid] = NO\_GROUP \quad \text{Ensure ea} \\
\quad\quad \land \text{ LET } newGroup \triangleq \\
\quad\quad\quad\quad [id \mapsto gid, \\
\quad\quad\quad\quad\ creator \ \mapsto actor, \\
\quad\quad\quad\quad\ members \mapsto [u \in USERS \mapsto \text{IF } u = actor \text{ THEN } 1 \text{ ELSE } 0], \\
\quad\quad\quad\quad\ expenseids \mapsto \{\}, \\
\quad\quad\quad\quad\ totalGifted \mapsto 0, \\
\quad\quad\quad\quad\ individualGiftsSent \mapsto [u \in USERS \mapsto 0]] \\
\quad\quad\quad\quad newReplica \triangleq \\
\quad\quad\quad\quad\quad [replicas[rid] \text{ EXCEPT } !.groups = [@ \text{ EXCEPT } ![gid] = newGroup]] \\
\quad\quad\quad\quad \text{IN} \quad \land\ replicas' = [replicas \text{ EXCEPT } ![rid] = newReplica] \\
\quad\quad\quad\quad\quad\quad \land\ actionCounter' = actionCounter + 1
\end{array}
$$

$$
\begin{array}{l}
AddMember \triangleq \\
\quad \exists\, actor,\ newMember \in USERS \quad : \\
\quad \exists\, gid \in POSSIBLE\_GROUP\_IDs : \\
\quad \exists\, rid \in POSSIBLE\_REPLICA\_IDs : \\
\quad\quad \land\ ASSIGNED\_REPLICA[actor] = rid \\
\quad\quad \land\ replicas[rid].groups[gid] \neq NO\_GROUP \\
\quad\quad \land\ replicas[rid].groups[gid].members[actor]\%2 = 1 \\
\quad\quad \land\ replicas[rid].groups[gid].members[newMember]\%2 = 0 \\
\quad\quad \land \text{ LET } newReplica \triangleq \\
\quad\quad\quad\quad [replicas[rid] \text{ EXCEPT } !.groups =
\end{array}
$$

$$
\begin{array}{l}
\qquad\qquad [@ \text{ EXCEPT } ![gid].members[newMember] = @ + 1]] \\
\quad \text{IN} \quad \wedge\, replicas' = [replicas \text{ EXCEPT } ![rid] = newReplica] \\
\qquad\quad \wedge\, actionCounter' = actionCounter + 1
\end{array}
$$

$$
\begin{array}{l}
LeaveGroup \triangleq \\
\quad \exists\, actor \in USERS : \\
\quad \exists\, gid \in POSSIBLE\_GROUP\_IDs : \\
\quad \exists\, rid \in POSSIBLE\_REPLICA\_IDs : \\
\qquad \wedge\, ASSIGNED\_REPLICA[actor] = rid \\
\qquad \wedge\, replicas[rid].groups[gid] \neq NO\_GROUP \\
\qquad \wedge\, replicas[rid].groups[gid].members[actor]\%2 = 1 \\
\qquad \wedge\, Balance(actor,\, gid,\, replicas[rid]) \geq 0 \\
\qquad \wedge \text{ LET } updatedGroups \triangleq \\
\qquad\qquad [replicas[rid].groups \text{ EXCEPT} \\
\qquad\qquad\quad ![gid].members[actor] = @ + 1] \\
\qquad\quad newGroups \triangleq RecalcGifts(updatedGroups,\, replicas[rid].recordedExpenses) \\
\qquad\quad newReplica \triangleq \\
\qquad\qquad [replicas[rid] \text{ EXCEPT } !.groups = newGroups] \\
\qquad \text{IN} \quad \wedge\, replicas' = [replicas \text{ EXCEPT } ![rid] = newReplica] \\
\qquad\quad \wedge\, actionCounter' = actionCounter + 1
\end{array}
$$

Expense actions

$$
\begin{array}{l}
CreateExpense \triangleq \\
\quad \exists\, actor,\, payer \in USERS : \\
\quad \exists\, shares \in POSSIBLE\_SHARES : \\
\quad \exists\, eid \in POSSIBLE\_EXPENSE\_IDs : \\
\quad \exists\, rid \in POSSIBLE\_REPLICA\_IDs \ : \\
\qquad \wedge\, ASSIGNED\_REPLICA[actor] = rid \\
\qquad \wedge\, SumFunction(shares) > 0 \\
\qquad \wedge\, \forall\, otherRid \in POSSIBLE\_REPLICA\_IDs : replicas[otherRid].recordedExpenses[eid] = NO\_EXPENS \\
\qquad \wedge \text{ LET } newExpense \triangleq \\
\qquad\qquad [id \mapsto eid, \\
\qquad\qquad\ group \mapsto NO\_GROUP, \\
\qquad\qquad\ groupCounter \mapsto [gid \in POSSIBLE\_GROUP\_IDs \mapsto 0], \\
\qquad\qquad\ version \mapsto 0, \\
\qquad\qquad\ payer \mapsto payer, \\
\qquad\qquad\ amount \mapsto SumFunction(shares), \\
\qquad\qquad\ shares \ \mapsto shares, \\
\qquad\qquad\ deleted \mapsto \text{FALSE}] \\
\qquad\quad newReplica \triangleq \\
\qquad\qquad [replicas[rid] \text{ EXCEPT } !.recordedExpenses = [@ \text{ EXCEPT } ![eid] = newExpense]] \\
\qquad \text{IN} \quad \wedge\, replicas' = [replicas \text{ EXCEPT } ![rid] = newReplica]
\end{array}
$$

$$\land\ actionCounter' = actionCounter + 1$$

$AddExpenseToGroup \triangleq$
$\exists\, actor \in USERS :$
$\exists\, eid \in POSSIBLE\_EXPENSE\_IDs :$
$\exists\, gid \in POSSIBLE\_GROUP\_IDs :$
$\exists\, rid \in POSSIBLE\_REPLICA\_IDs :$
$\land\ ASSIGNED\_REPLICA[actor] = rid$
$\land\ replicas[rid].groups[gid] \neq NO\_GROUP$
$\land\ replicas[rid].recordedExpenses[eid] \neq NO\_EXPENSE$
$\land\ replicas[rid].groups[gid].members[actor]\%2 = 1$
$\land\ replicas[rid].recordedExpenses[eid].payer = actor$
$\land\ replicas[rid].recordedExpenses[eid].group = NO\_GROUP$
$\land\ replicas[rid].groups[gid].members[replicas[rid].recordedExpenses[eid].payer]\%2 = 1$
$\land\ \{u \in USERS : replicas[rid].recordedExpenses[eid].shares[u] > 0\}$
$\quad \subseteq \{u \in USERS : replicas[rid].groups[gid].members[u]\%2 = 1\}$
$\land$ LET $newExpense \triangleq$
$[replicas[rid].recordedExpenses[eid]$ EXCEPT $!.group = gid,\ !.groupCounter[gid] = @ + 1]$
$newExpenses \triangleq$
$[replicas[rid].recordedExpenses$ EXCEPT $![eid] = newExpense]$
$newGroups \triangleq$
$[replicas[rid].groups$ EXCEPT $![gid].expenseids = @ \cup \{eid\}]$
$newReplica \triangleq$
$[replicas[rid]$ EXCEPT $!.recordedExpenses = newExpenses,$
$!.groups = newGroups]$
IN $\land\ replicas' = [replicas$ EXCEPT $![rid] = newReplica]$
$\land\ actionCounter' = actionCounter + 1$

$RemoveExpenseFromGroup \triangleq$
$\exists\, actor \in USERS :$
$\exists\, eid \in POSSIBLE\_EXPENSE\_IDs :$
$\exists\, gid \in POSSIBLE\_GROUP\_IDs :$
$\exists\, rid \in POSSIBLE\_REPLICA\_IDs :$
$\land\ ASSIGNED\_REPLICA[actor] = rid$
$\land\ replicas[rid].groups[gid] \neq NO\_GROUP$
$\land\ replicas[rid].recordedExpenses[eid] \neq NO\_EXPENSE$
$\land\ replicas[rid].groups[gid].members[actor]\%2 = 1$
$\land\ replicas[rid].recordedExpenses[eid].group = gid$
$\land\ replicas[rid].recordedExpenses[eid].payer = actor$
$\land$ LET $newExpense \triangleq [replicas[rid].recordedExpenses[eid]$ EXCEPT $!.group = NO\_GROUP,\ !.groupC$
$newExpenses \triangleq [replicas[rid].recordedExpenses$ EXCEPT $![eid] = newExpense]$
$groupsWithout \triangleq [replicas[rid].groups$ EXCEPT $![gid].expenseids = @ \setminus \{eid\}]$
$newGroups \triangleq RecalcGifts(groupsWithout,\ newExpenses)$
$newReplica \triangleq [replicas[rid]$ EXCEPT $!.recordedExpenses = newExpenses,$
$!.groups = newGroups]$

```
        IN    ∧ replicas′ = [replicas EXCEPT ![rid] = newReplica]
              ∧ actionCounter′ = actionCounter + 1

ModifyExpenseParameters ≜
    ∃ actor ∈ USERS :
    ∃ shares ∈ POSSIBLE_SHARES :
    ∃ eid ∈ POSSIBLE_EXPENSE_IDs :
    ∃ rid ∈ POSSIBLE_REPLICA_IDs :
      ∧ ASSIGNED_REPLICA[actor] = rid
      ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
      ∧ replicas[rid].recordedExpenses[eid].payer = actor
      ∧ SumFunction(shares) > 0
      ∧ IF replicas[rid].recordedExpenses[eid].group ≠ NO_GROUP
        THEN {u ∈ USERS : shares[u] > 0}
                ⊆ {u ∈ USERS : replicas[rid].groups[replicas[rid].recordedExpenses[eid].group].members[u]%
        ELSE TRUE
      ∧ LET newExpenses ≜
                [replicas[rid].recordedExpenses EXCEPT
                  ![eid].shares  = shares,
                  ![eid].amount = SumFunction(shares),
                  ![eid].version = @ + 1]
            newGroups ≜
              IF replicas[rid].recordedExpenses[eid].group = NO_GROUP
               THEN replicas[rid].groups
               ELSE RecalcGifts(replicas[rid].groups, newExpenses)
            newReplica ≜
              [replicas[rid] EXCEPT !.recordedExpenses = newExpenses,
                                    !.groups = newGroups]
        IN    ∧ replicas′ = [replicas EXCEPT ![rid] = newReplica]
              ∧ actionCounter′ = actionCounter + 1

DeleteExpense ≜
    ∃ actor ∈ USERS :
    ∃ eid ∈ POSSIBLE_EXPENSE_IDs :
    ∃ rid ∈ POSSIBLE_REPLICA_IDs :
      ∧ ASSIGNED_REPLICA[actor] = rid
      ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
      ∧ replicas[rid].recordedExpenses[eid].payer = actor
      ∧ replicas[rid].recordedExpenses[eid].deleted = FALSE
      ∧ IF replicas[rid].recordedExpenses[eid].group ≠ NO_GROUP
        THEN ∧ replicas[rid].groups[replicas[rid].recordedExpenses[eid].group].members[actor]%2 = 1
        ELSE TRUE
      ∧ LET newExpenses ≜
                [replicas[rid].recordedExpenses EXCEPT ![eid].deleted = TRUE]
            newGroups ≜
```

$RecalcGifts(replicas[rid].groups,\ newExpenses)$
$newReplica \triangleq$
$[replicas[rid]$ EXCEPT $!.recordedExpenses = newExpenses,$
$!.groups = newGroups]$
IN $\land\ replicas' = [replicas$ EXCEPT $![rid] = newReplica]$
$\land\ actionCounter' = actionCounter + 1$

Merge relation

$MergeReplicas \triangleq$
$\exists\, ownRid,\ otherRid \in POSSIBLE\_REPLICA\_IDs :$
$\land\ ownRid \neq otherRid$
$\land$ LET

first: merge expenses so groups can consult the merged expense state
$mergedExpenses \triangleq$
$[eid \in POSSIBLE\_EXPENSE\_IDs \mapsto$
IF $replicas[ownRid].recordedExpenses[eid] = NO\_EXPENSE \land replicas[otherRid].recordedExpenses$
THEN $NO\_EXPENSE$
ELSE IF $replicas[ownRid].recordedExpenses[eid] \neq NO\_EXPENSE \land replicas[otherRid].recordedE$
THEN $replicas[ownRid].recordedExpenses[eid]$
ELSE IF $replicas[ownRid].recordedExpenses[eid] = NO\_EXPENSE \land replicas[otherRid].recordedE$
THEN $replicas[otherRid].recordedExpenses[eid]$
ELSE
LET $expOwn \triangleq replicas[ownRid].recordedExpenses[eid]$
$expOther \triangleq replicas[otherRid].recordedExpenses[eid]$
$mergedDeleted \triangleq expOwn.deleted \lor expOther.deleted$
$mergedVersion \triangleq$
CHOOSE $v \in \{expOwn.version,\ expOther.version\} :$
$v \geq expOwn.version \land v \geq expOther.version$
$useOwnVersion \triangleq expOwn.version \geq expOther.version$
$baseExp \triangleq$ IF $useOwnVersion$ THEN $expOwn$ ELSE $expOther$
$mergedGroupCounter \triangleq$
$[g \in POSSIBLE\_GROUP\_IDs \mapsto$
CHOOSE $n \in \{expOwn.groupCounter[g],\ expOther.groupCounter[g]\} :$
$n \geq expOwn.groupCounter[g] \land n \geq expOther.groupCounter[g]]$
$mergedGroup \triangleq$
IF $\exists\, g \in POSSIBLE\_GROUP\_IDs : mergedGroupCounter[g]\%2 = 1$
THEN CHOOSE $g \in POSSIBLE\_GROUP\_IDs : mergedGroupCounter[g]\%2 = 1$
ELSE NO_GROUP
IN $[baseExp$ EXCEPT
$!.deleted = mergedDeleted,$
$!.groupCounter = mergedGroupCounter,$

```
                !.group = mergedGroup,
                !.version = mergedVersion]
    ]

  then: merge groups, but compute expenseids from mergedExpenses to stay consistent
  mergedGroups ≜
    [gid ∈ POSSIBLE_GROUP_IDs ↦
      IF replicas[ownRid].groups[gid] = NO_GROUP ∧ replicas[otherRid].groups[gid] = NO_GROUP
      THEN NO_GROUP
      ELSE IF replicas[ownRid].groups[gid] ≠ NO_GROUP ∧ replicas[otherRid].groups[gid] = NO_GR
      THEN replicas[ownRid].groups[gid]
      ELSE IF replicas[ownRid].groups[gid] = NO_GROUP ∧ replicas[otherRid].groups[gid] ≠ NO_GR
      THEN replicas[otherRid].groups[gid]
      ELSE
       LET grpOwn ≜ replicas[ownRid].groups[gid]
           grpOther ≜ replicas[otherRid].groups[gid]
           mergedMembers ≜
             [u ∈ USERS ↦
               CHOOSE n ∈ {grpOwn.members[u], grpOther.members[u]} :
                 n ≥ grpOwn.members[u] ∧ n ≥ grpOther.members[u]]
         take union of expense ids, but filter by mergedExpenses so only expenses that actually belong to this grou
           mergedExpenseIds ≜
             {eid ∈ (grpOwn.expenseids ∪ grpOther.expenseids) :
                 ∧ mergedExpenses[eid] ≠ NO_EXPENSE
                 ∧ mergedExpenses[eid].group = gid
                 ∧ mergedExpenses[eid].deleted = FALSE}
           mergedGroup ≜ [grpOwn EXCEPT
                            !.members = mergedMembers,
                            !.expenseids = mergedExpenseIds]
           balances ≜ ComputeBalances(mergedGroup, mergedExpenses)
       IN  ComputeGifts(mergedGroup, balances)
    ]

  newReplica ≜
    [replicas[ownRid] EXCEPT
      !.groups = mergedGroups,
      !.recordedExpenses = mergedExpenses]

IN  replicas′ = [replicas EXCEPT ![ownRid] = newReplica]
    ∧ UNCHANGED actionCounter
```

Next relation

$Next \triangleq$

$\lor$ *CreateGroup*
$\lor$ *AddMember*
$\lor$ *LeaveGroup*
$\lor$ *CreateExpense*
$\lor$ *AddExpenseToGroup*
$\lor$ *RemoveExpenseFromGroup*
$\lor$ *ModifyExpenseParameters*
$\lor$ *DeleteExpense*
$\lor$ *MergeReplicas*
$\lor$ UNCHANGED $\langle replicas,\ actionCounter \rangle$

Specification

$Spec \triangleq Init \land \Box[Next]_{\langle replicas,\ actionCounter \rangle} \land \mathrm{WF}_{\langle replicas,\ actionCounter \rangle}(MergeReplicas)$

Invariants

$TypeOK \triangleq$
$\forall\, rid \in POSSIBLE\_REPLICA\_IDs :$
$\land\ replicas[rid].recordedExpenses$
$\in [POSSIBLE\_EXPENSE\_IDs \to (Expense \cup \{NO\_EXPENSE\})]$
$\land\ replicas[rid].groups$
$\in [POSSIBLE\_GROUP\_IDs \to (Group \cup \{NO\_GROUP\})]$

$Inv\_Conservation\_of\_amount \triangleq$
$\forall\, rid \in POSSIBLE\_REPLICA\_IDs :$
$\forall\, eid \in POSSIBLE\_EXPENSE\_IDs :$
$replicas[rid].recordedExpenses[eid] \neq NO\_EXPENSE \Rightarrow$
LET $e \triangleq replicas[rid].recordedExpenses[eid]$
IN $e.amount = SumFunction(e.shares)$

$Inv\_GroupContainsConsistentEids \triangleq$
$\forall\, rid \in POSSIBLE\_REPLICA\_IDs :$
$\forall\, gid \in POSSIBLE\_GROUP\_IDs :$
$replicas[rid].groups[gid] \neq NO\_GROUP \Rightarrow$
$\forall\, eid \in replicas[rid].groups[gid].expenseids :$
$replicas[rid].recordedExpenses[eid].group = gid$

$Inv\_GroupBalanceZero \triangleq$
$\forall\, rid \in POSSIBLE\_REPLICA\_IDs :$
$\forall\, gid \in POSSIBLE\_GROUP\_IDs :$
$replicas[rid].groups[gid] \neq NO\_GROUP \Rightarrow$

LET $allUsers \triangleq$
$\{u \in USERS : replicas[rid].groups[gid].members[u]\%2 = 1 \lor replicas[rid].groups[gid].members[u] > 0\}$
$\{u \in USERS : replicas[rid].groups[gid].members[u] > 0\}$ include every user that was a member of the
$total \triangleq$
$SumFunction([u \in allUsers \mapsto Balance(u, gid, replicas[rid])])$
IN $total + replicas[rid].groups[gid].totalGifted = 0$

$Inv \triangleq$
$\land\ TypeOK$
$\land\ Inv\_Conservation\_of\_amount$
$\land\ Inv\_GroupContainsConsistentEids$
$\land\ Inv\_GroupBalanceZero$

Liveness

$ExpenseSameAcrossReplicas(eid) \triangleq$ Helper to check if an expense is the same on two replicas
$\forall\, r1, r2 \in POSSIBLE\_REPLICA\_IDs :$
$replicas[r1].recordedExpenses[eid] = replicas[r2].recordedExpenses[eid]$

$GroupSameAcrossReplicas(gid) \triangleq$ Helper to check if a group is the same on two replicas
$\forall\, r1, r2 \in POSSIBLE\_REPLICA\_IDs :$
$replicas[r1].groups[gid] = replicas[r2].groups[gid]$

$GroupMembersSameAcrossReplicas(gid) \triangleq$ Helper to check if the members of a group are the same on two replicas
$\forall\, r1, r2 \in POSSIBLE\_REPLICA\_IDs :$
$replicas[r1].groups[gid] \neq NO\_GROUP \land replicas[r2].groups[gid] \neq NO\_GROUP \Rightarrow$
$replicas[r1].groups[gid].members = replicas[r2].groups[gid].members$

$Liveness\_ExpenseCreationPropagates \triangleq$ Ensures a created expense gets propagated to all replicas
$\forall\, eid \in POSSIBLE\_EXPENSE\_IDs :$
$(\exists\, rid \in POSSIBLE\_REPLICA\_IDs :$
$replicas[rid].recordedExpenses[eid] \neq NO\_EXPENSE)$
$\rightsquigarrow$
$ExpenseSameAcrossReplicas(eid)$

$Liveness\_ExpenseModificationPropagates \triangleq$ Ensures that expense modifications get propagated to all replicas. Checks
$\forall\, eid \in POSSIBLE\_EXPENSE\_IDs :$
$(\exists\, rid1, rid2 \in POSSIBLE\_REPLICA\_IDs :$
$\land\ replicas[rid1].recordedExpenses[eid] \neq NO\_EXPENSE$
$\land\ replicas[rid2].recordedExpenses[eid] \neq NO\_EXPENSE$
$\land\ replicas[rid1].recordedExpenses[eid] \neq replicas[rid2].recordedExpenses[eid])$
$\rightsquigarrow$

$ExpenseSameAcrossReplicas(eid)$

$Liveness\_GroupCreationPropagates \triangleq$ Ensure a created group gets propagated to all replicas

$\forall\, gid \in POSSIBLE\_GROUP\_IDs :$
$(\exists\, rid \in POSSIBLE\_REPLICA\_IDs :$
$replicas[rid].groups[gid] \neq NO\_GROUP)$
$\rightsquigarrow$
$GroupSameAcrossReplicas(gid)$

$Liveness\_GroupMembershipPropagates \triangleq$ Ensures user group membership changes get propagated to all replicas

$\forall\, gid \in POSSIBLE\_GROUP\_IDs :$
$(\exists\, rid1,\, rid2 \in POSSIBLE\_REPLICA\_IDs :$
$\land\, replicas[rid1].groups[gid] \neq NO\_GROUP$
$\land\, replicas[rid2].groups[gid] \neq NO\_GROUP$
$\land\, replicas[rid1].groups[gid].members \neq replicas[rid2].groups[gid].members)$
$\rightsquigarrow$
$GroupMembersSameAcrossReplicas(gid)$

* Modification History
* Last modified *Mon Oct* 27 14:28:57 *CET* 2025 by *floyd*
* Created *Fri Oct* 24 11:14:17 *CEST* 2025 by *floyd*