```
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 ≜
        [id : POSSIBLE_EXPENSE_IDs,
         group : POSSIBLE_GROUP_IDs ∪ {NO_GROUP},
         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,
         shares : [USERS → Nat],   if payer absorbs the share of a left member, their share can be higher than the max in
         acknowledged_shares : [USERS → BOOLEAN ],
         deleted : BOOLEAN ]

Group ≜
        [id : POSSIBLE_GROUP_IDs,
         members : [USERS → Nat],   Casul length counter for each user
         totalGifted : Nat,
         individualGiftsSent : [USERS → Nat]]

Replica ≜
       [id : POSSIBLE_REPLICA_IDs,
        recordedExpenses : [POSSIBLE_EXPENSE_IDs → (Expense ∪ {NO_EXPENSE})],
        groups : [POSSIBLE_GROUP_IDs → (Group ∪ {NO_GROUP})]
       ]

\* ---------------------
\* Initialization
\* ---------------------
Init ≜
      ∧ replicas =
```

```
48              [rid ∈ POSSIBLE_REPLICA_IDs ↦
49                 [id ↦ rid,
50                  recordedExpenses ↦ [eid ∈ POSSIBLE_EXPENSE_IDs ↦ NO_EXPENSE],
51                  groups ↦ [gid ∈ POSSIBLE_GROUP_IDs ↦ NO_GROUP]
52                 ]
53              ]
54        ∧ actionCounter = 0

56  ----------------------------
57  Helper Functions
58  ----------------------------

60  Get expenseIds that are added to a specific group
61  GroupExpenseIds(gid, recordedExpensesIn) ≜
62    {eid ∈ POSSIBLE_EXPENSE_IDs :
63        ∧ recordedExpensesIn[eid] ≠ NO_EXPENSE
64        ∧ recordedExpensesIn[eid].deleted = FALSE
65        ∧ recordedExpensesIn[eid].group = gid
66        ∧ ∀ u ∈ DOMAIN recordedExpensesIn[eid].shares :
67            (recordedExpensesIn[eid].shares[u] > 0)
68              ⇒ recordedExpensesIn[eid].acknowledged_shares[u] = TRUE}

70  IsMember(memberCounter) ≜
71    memberCounter%2 = 1

73  WasEverMember(memberCounter) ≜
74    memberCounter > 0

76  RECURSIVE SumFunction(_)
77  SumFunction(F) ≜
78        IF DOMAIN F = {} THEN 0
79         ELSE LET d ≜ CHOOSE x ∈ DOMAIN F : TRUE
80              IN   F[d] + SumFunction([y ∈ DOMAIN F \ {d} ↦ F[y]])


83  Balance(u, gid, replica) ≜
84          LET groupExpenses ≜ GroupExpenseIds(gid, replica.recordedExpenses)
85          IN   SumFunction([eid ∈ groupExpenses ↦
86                                IF replica.recordedExpenses[eid].payer = u
87                                 THEN replica.recordedExpenses[eid].amount ELSE 0])
88                - SumFunction([eid ∈ groupExpenses ↦ replica.recordedExpenses[eid].shares[u]])
89                - replica.groups[gid].individualGiftsSent[u]


92  ComputeBalances(grp, recordedExpensesIn) ≜
93        [u ∈ USERS ↦
94          LET groupExpenses ≜ GroupExpenseIds(grp.id, recordedExpensesIn)
95          IN   SumFunction([eid ∈ groupExpenses ↦
```

```
                                IF recordedExpensesIn[eid].payer = u
                                 THEN recordedExpensesIn[eid].amount ELSE 0])
            − SumFunction([eid ∈ groupExpenses ↦ recordedExpensesIn[eid].shares[u]])
    ]

ComputeGifts(grp, balances) ≜
    LET giftingUsers ≜
          {u ∈ USERS : ¬IsMember(grp.members[u]) ∧ balances[u] > 0}
        newTotalGifted ≜ SumFunction([u ∈ giftingUsers ↦ balances[u]])
        newIndividualGifts ≜ [u ∈ USERS ↦ IF u ∈ giftingUsers THEN balances[u] ELSE 0]
    IN  [grp EXCEPT !.totalGifted = newTotalGifted,
                    !.individualGiftsSent = newIndividualGifts]

RecalcGifts(groupsIn, recordedExpensesIn) ≜
    [gid ∈ POSSIBLE_GROUP_IDs ↦
      IF groupsIn[gid] = NO_GROUP THEN NO_GROUP
       ELSE LET grp ≜ groupsIn[gid]
                balances ≜ ComputeBalances(grp, recordedExpensesIn)
            IN  ComputeGifts(grp, balances)
    ]

(* ---------------------- *)
(* Group actions          *)
(* ---------------------- *)

CreateGroup ≜
    ∃ actor ∈ USERS :
    ∃ gid ∈ POSSIBLE_GROUP_IDs :
    ∃ rid ∈ POSSIBLE_REPLICA_IDs :
      ∧ ASSIGNED_REPLICA[actor] = rid
        (* Ensure each gid is only used once across replicas, real app use pseudorandom functions or similar *)
      ∧ ∀ otherRid ∈ POSSIBLE_REPLICA_IDs : replicas[otherRid].groups[gid] = NO_GROUP
      ∧ LET newGroup ≜
            [id ↦ gid,
             members ↦ [u ∈ USERS ↦ IF u = actor THEN 1 ELSE 0],
             totalGifted ↦ 0,
             individualGiftsSent ↦ [u ∈ USERS ↦ 0]]
          newReplica ≜
            [replicas[rid] EXCEPT !.groups = [@ EXCEPT ![gid] = newGroup]]
        IN  ∧ replicas' = [replicas EXCEPT ![rid] = newReplica]
            ∧ actionCounter' = actionCounter + 1

AddMember ≜
    ∃ actor, newMember ∈ USERS    :
```

```
143         ∃ gid ∈ POSSIBLE_GROUP_IDs :
144         ∃ rid ∈ POSSIBLE_REPLICA_IDs :
145           ∧ ASSIGNED_REPLICA[actor] = rid
146           ∧ replicas[rid].groups[gid] ≠ NO_GROUP
147           ∧ IsMember(replicas[rid].groups[gid].members[actor])
148           ∧ ¬IsMember(replicas[rid].groups[gid].members[newMember])
149           ∧ LET newReplica ≜
150                   [replicas[rid] EXCEPT !.groups =
151                     [@ EXCEPT ![gid].members[newMember] = @ + 1]]
152               IN  ∧ replicas′ = [replicas EXCEPT ![rid] = newReplica]
153                   ∧ actionCounter′ = actionCounter + 1

155 LeaveGroup ≜
156         ∃ actor ∈ USERS :
157         ∃ gid ∈ POSSIBLE_GROUP_IDs :
158         ∃ rid ∈ POSSIBLE_REPLICA_IDs :
159           ∧ ASSIGNED_REPLICA[actor] = rid
160           ∧ replicas[rid].groups[gid] ≠ NO_GROUP
161           ∧ IsMember(replicas[rid].groups[gid].members[actor])
162           ∧ Balance(actor, gid, replicas[rid]) ≥ 0
163           ∧ LET updatedGroups ≜
164                   [replicas[rid].groups EXCEPT
165                     ![gid].members[actor] = @ + 1]
166                 newGroups ≜ RecalcGifts(updatedGroups, replicas[rid].recordedExpenses)
167                 newReplica ≜
168                   [replicas[rid] EXCEPT !.groups = newGroups]
169               IN  ∧ replicas′ = [replicas EXCEPT ![rid] = newReplica]
170                   ∧ actionCounter′ = actionCounter + 1


173 ----------------------------
174 Expense actions
175 ----------------------------

177 CreateExpense ≜
178         ∃ actor ∈ USERS :
179         ∃ shares ∈ POSSIBLE_SHARES :
180         ∃ eid ∈ POSSIBLE_EXPENSE_IDs :
181         ∃ rid ∈ POSSIBLE_REPLICA_IDs :
182           ∧ ASSIGNED_REPLICA[actor] = rid
183           ∧ SumFunction(shares) > 0
184           ∧ ∀ otherRid ∈ POSSIBLE_REPLICA_IDs : replicas[otherRid].recordedExpenses[eid] = NO_EXPENS
185           ∧ LET newExpense ≜
186                   [id ↦ eid,
187                    group ↦ NO_GROUP,
188                    version ↦ 0,
```

```
                payer ↦ actor,
                amount ↦ SumFunction(shares),
                shares  ↦ shares,
                acknowledged_shares ↦ [u ∈ USERS ↦ IF u = actor ∧ shares[u] > 0 THEN TRUE ELSE FALSE],
                deleted ↦ FALSE]
              newReplica ≜
                [replicas[rid] EXCEPT !.recordedExpenses = [@ EXCEPT ![eid] = newExpense]]
            IN   ∧ replicas' = [replicas EXCEPT ![rid] = newReplica]
                 ∧ actionCounter' = actionCounter + 1

AddExpenseToGroup ≜
    ∃ actor ∈ USERS :
    ∃ eid ∈ POSSIBLE_EXPENSE_IDs :
    ∃ gid ∈ POSSIBLE_GROUP_IDs :
    ∃ rid ∈ POSSIBLE_REPLICA_IDs :
      ∧ ASSIGNED_REPLICA[actor] = rid
      ∧ replicas[rid].groups[gid] ≠ NO_GROUP
      ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
      ∧ IsMember(replicas[rid].groups[gid].members[actor])
      ∧ replicas[rid].recordedExpenses[eid].payer = actor
      ∧ replicas[rid].recordedExpenses[eid].group = NO_GROUP
      ∧ {u ∈ USERS : replicas[rid].recordedExpenses[eid].shares[u] > 0}
            ⊆ {u ∈ USERS : IsMember(replicas[rid].groups[gid].members[u])}
      ∧ LET newExpense ≜
              [replicas[rid].recordedExpenses[eid] EXCEPT !.group = gid, !.version = @ + 1]
            newExpenses ≜
              [replicas[rid].recordedExpenses EXCEPT ![eid] = newExpense]
            newReplica ≜
              [replicas[rid] EXCEPT !.recordedExpenses = newExpenses]
        IN   ∧ replicas' = [replicas EXCEPT ![rid] = newReplica]
             ∧ actionCounter' = actionCounter + 1

RemoveExpenseFromGroup ≜
    ∃ actor ∈ USERS :
    ∃ eid ∈ POSSIBLE_EXPENSE_IDs :
    ∃ gid ∈ POSSIBLE_GROUP_IDs :
    ∃ rid ∈ POSSIBLE_REPLICA_IDs :
      ∧ ASSIGNED_REPLICA[actor] = rid
      ∧ replicas[rid].groups[gid] ≠ NO_GROUP
      ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
      ∧ IsMember(replicas[rid].groups[gid].members[actor])
      ∧ replicas[rid].recordedExpenses[eid].group = gid
      ∧ replicas[rid].recordedExpenses[eid].payer = actor
      ∧ LET newExpense ≜ [replicas[rid].recordedExpenses[eid] EXCEPT !.group = NO_GROUP,
                                                                     !.version = @ + 1,
```

```
                                    !.acknowledged_shares = [u ∈ USERS ↦ IF u = actor ∧ replicas[rid].recorded
            newExpenses ≜ [replicas[rid].recordedExpenses EXCEPT ![eid] = newExpense]
            newGroups ≜ RecalcGifts(replicas[rid].groups, newExpenses)
            newReplica ≜ [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
      ∧ ¬replicas[rid].recordedExpenses[eid].deleted
      ∧ IF replicas[rid].recordedExpenses[eid].group ≠ NO_GROUP
        THEN {u ∈ USERS : shares[u] > 0}
             ⊆ {u ∈ USERS : IsMember(replicas[rid].groups[replicas[rid].recordedExpenses[eid].group].m
        ELSE TRUE
      ∧ LET newExpenses ≜
              [replicas[rid].recordedExpenses EXCEPT
                ![eid].shares = shares,
                ![eid].acknowledged_shares = [u ∈ USERS ↦ IF u = actor ∧ shares[u] > 0 THEN TRUE ELSE
                ![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 ∧ IsMember(replicas[rid].groups[replicas[rid].recordedExpenses[eid].group].members[actor])
          ELSE TRUE
        ∧ LET newExpenses ≜
                [replicas[rid].recordedExpenses EXCEPT ![eid].deleted = TRUE, ![eid].version = @ + 1]
            newGroups ≜
              RecalcGifts(replicas[rid].groups, newExpenses)
            newReplica ≜
              [replicas[rid] EXCEPT !.recordedExpenses = newExpenses,
                                    !.groups = newGroups]
          IN  ∧ replicas' = [replicas EXCEPT ![rid] = newReplica]
              ∧ actionCounter' = actionCounter + 1

AcknowledgeShare ≜
  ∃ actor ∈ USERS :
  ∃ eid ∈ POSSIBLE_EXPENSE_IDs :
  ∃ rid ∈ POSSIBLE_REPLICA_IDs :
    ∧ ASSIGNED_REPLICA[actor] = rid
    ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
    ∧ replicas[rid].recordedExpenses[eid].deleted = FALSE
    ∧ replicas[rid].recordedExpenses[eid].group ≠ NO_GROUP
    ∧ IsMember(replicas[rid].groups[replicas[rid].recordedExpenses[eid].group].members[actor])
    ∧ replicas[rid].recordedExpenses[eid].shares[actor] > 0
    ∧ replicas[rid].recordedExpenses[eid].acknowledged_shares[actor] = FALSE
    ∧ LET newExpenses ≜
            [replicas[rid].recordedExpenses EXCEPT ![eid].acknowledged_shares[actor] = TRUE]
        newGroups ≜
          RecalcGifts(replicas[rid].groups, newExpenses)
        newReplica ≜
          [replicas[rid] EXCEPT !.recordedExpenses = newExpenses,
                                !.groups = newGroups]
      IN  ∧ replicas' = [replicas EXCEPT ![rid] = newReplica]
          ∧ actionCounter' = actionCounter + 1
          ∧ UNCHANGED actionCounter


 Payer absorbs share of a member who left and never acknowledged
PayerAbsorbsLeftMemberShare ≜
  ∃ actor ∈ USERS :
  ∃ eid ∈ POSSIBLE_EXPENSE_IDs :
  ∃ leftMember ∈ USERS :
  ∃ rid ∈ POSSIBLE_REPLICA_IDs :
    ∧ ASSIGNED_REPLICA[actor] = rid
    ∧ actor ≠ leftMember
    ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
```

```
        ∧ replicas[rid].recordedExpenses[eid].deleted = FALSE
        ∧ replicas[rid].recordedExpenses[eid].payer = actor
        ∧ replicas[rid].recordedExpenses[eid].shares[leftMember] > 0
        ∧ replicas[rid].recordedExpenses[eid].acknowledged_shares[leftMember] = FALSE
        ∧ replicas[rid].recordedExpenses[eid].group ≠ NO_GROUP
        ∧ LET gid ≜ replicas[rid].recordedExpenses[eid].group
          IN  ∧ replicas[rid].groups[gid] ≠ NO_GROUP
              ∧ IsMember(replicas[rid].groups[gid].members[actor])
              ∧ ¬IsMember(replicas[rid].groups[gid].members[leftMember])
              ∧ WasEverMember(replicas[rid].groups[gid].members[leftMember])
              ∧ LET oldShares ≜ replicas[rid].recordedExpenses[eid].shares
                    leftShare ≜ oldShares[leftMember]
                    newShares ≜ [u ∈ USERS ↦
                      IF u = actor THEN oldShares[u] + leftShare
                       ELSE IF u = leftMember THEN 0
                       ELSE oldShares[u]]
                    newExpenses ≜
                      [replicas[rid].recordedExpenses EXCEPT
                        ![eid].shares = newShares,
                        ![eid].acknowledged_shares = [u ∈ USERS ↦
                          IF u = leftMember THEN FALSE
                          ELSE replicas[rid].recordedExpenses[eid].acknowledged_shares[u]],
                        ![eid].version = @ + 1]
                    newGroups ≜ RecalcGifts(replicas[rid].groups, newExpenses)
                    newReplica ≜
                      [replicas[rid] EXCEPT !.recordedExpenses = newExpenses,
                                            !.groups = newGroups]
                IN  ∧ replicas′ = [replicas EXCEPT ![rid] = newReplica]
                    ∧ UNCHANGED actionCounter

    Alternative conflict resolution: member rejoins to acknowledge
    Follows the add member protocoll by having a group member reentering the rejoining one
RejoinToAcknowledge ≜
    ∃ inviter, rejoiner ∈ USERS :
    ∃ gid ∈ POSSIBLE_GROUP_IDs :
    ∃ rid ∈ POSSIBLE_REPLICA_IDs :
        ∧ ASSIGNED_REPLICA[inviter] = rid
        ∧ replicas[rid].groups[gid]          ≠ NO_GROUP
        ∧ IsMember(replicas[rid].groups[gid].members[inviter])
        ∧ ¬IsMember(replicas[rid].groups[gid].members[rejoiner])
        ∧ WasEverMember(replicas[rid].groups[gid].members[rejoiner])
        ∧ ∃ eid ∈ POSSIBLE_EXPENSE_IDs :
            ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
            ∧ replicas[rid].recordedExpenses[eid].group = gid
            ∧ replicas[rid].recordedExpenses[eid].shares[rejoiner] > 0
```

```
                ∧ replicas[rid].recordedExpenses[eid].acknowledged_shares[rejoiner] = FALSE
          ∧ LET newReplica ≜
                  [replicas[rid] EXCEPT !.groups =
                    [@ EXCEPT ![gid].members[rejoiner] = @ + 1]]
            IN    ∧ replicas' = [replicas EXCEPT ![rid]    = newReplica]
                  ∧ UNCHANGED actionCounter


\* ---------------------------
\* Merge action helpers
\* ---------------------------

MergeExpense(expOwn, expOther) ≜
      IF expOwn = NO_EXPENSE ∧ expOther = NO_EXPENSE
       THEN NO_EXPENSE
       ELSE IF expOwn ≠ NO_EXPENSE ∧ expOther = NO_EXPENSE
       THEN expOwn
       ELSE IF expOwn = NO_EXPENSE ∧ expOther ≠ NO_EXPENSE
       THEN expOther
       ELSE IF expOwn.version > expOther.version
       THEN expOwn
       ELSE IF expOwn.version < expOther.version
       THEN expOther
       ELSE
        LET mergedAcknowledgedShares ≜
              [u ∈ USERS ↦
                expOwn.acknowledged_shares[u] ∨ expOther.acknowledged_shares[u]]
        IN  [expOwn EXCEPT !.acknowledged_shares = mergedAcknowledgedShares]
         LET mergedDeleted ≜ expOwn.deleted ∨ expOther.deleted
           useOwnVersion ≜ expOwn.version ≥ expOther.version
            baseExp ≜ IF useOwnVersion THEN expOwn ELSE expOther
              mergedGroup ≜ baseExp.group
                IF expOwn.group ≠ NO_GROUP THEN expOwn.group ELSE expOther.group
        IN  [ baseExp EXCEPT
              !.deleted = mergedDeleted,
               !.group = mergedGroup,
                !.version = IF useOwnVersion THEN expOwn.version ELSE expOther.version ]


MergeGroup(grpOwn, grpOther, mergedExpenses, gid) ≜
      IF grpOwn = NO_GROUP ∧ grpOther = NO_GROUP
       THEN NO_GROUP
       ELSE IF grpOwn ≠ NO_GROUP ∧ grpOther = NO_GROUP
       THEN grpOwn
       ELSE IF grpOwn = NO_GROUP ∧ grpOther ≠ NO_GROUP
       THEN grpOther
```

```
        ELSE
         LET mergedMembers ≜
                [u ∈ USERS ↦ CHOOSE n ∈ {grpOwn.members[u], grpOther.members[u]} :
                                n ≥ grpOwn.members[u] ∧ n ≥ grpOther.members[u]]
             mergedGroup ≜ [grpOwn EXCEPT !.members = mergedMembers]
             balances ≜ ComputeBalances(mergedGroup, mergedExpenses)
         IN  ComputeGifts(mergedGroup, balances)

\* ----------------------------
\* Merge action
\* ----------------------------

MergeReplicas ≜
 ∃ ownRid, otherRid ∈ POSSIBLE_REPLICA_IDs :
   ∧ ownRid ≠ otherRid
   ∧ LET
       mergedExpenses ≜
         [eid ∈ POSSIBLE_EXPENSE_IDs ↦
            MergeExpense(replicas[ownRid].recordedExpenses[eid],
                         replicas[otherRid].recordedExpenses[eid])]

       mergedGroups ≜
         [gid ∈ POSSIBLE_GROUP_IDs ↦
            MergeGroup(replicas[ownRid].groups[gid],
                       replicas[otherRid].groups[gid],
                       mergedExpenses,
                       gid)]

       newReplica ≜
         [replicas[ownRid] EXCEPT
            !.groups = mergedGroups,
            !.recordedExpenses = mergedExpenses]
     IN  replicas' = [replicas EXCEPT ![ownRid] = newReplica]
         ∧ UNCHANGED actionCounter

\* ----------------------------
\* Next relation
\* ----------------------------
Next ≜
   ∨ CreateGroup
   ∨ AddMember
   ∨ LeaveGroup
   ∨ CreateExpense
   ∨ AddExpenseToGroup
   ∨ RemoveExpenseFromGroup
   ∨ ModifyExpenseParameters
```

```
463       ∨ DeleteExpense
464       ∨ AcknowledgeShare
465       ∨ PayerAbsorbsLeftMemberShare
466       ∨ RejoinToAcknowledge
467       ∨ MergeReplicas
468       ∨ UNCHANGED ⟨replicas, actionCounter⟩


473   ----------------------------
474   Invariants
475   ----------------------------
476 TypeOK ≜
477   ∀ rid ∈ POSSIBLE_REPLICA_IDs :
478     ∧ replicas[rid].recordedExpenses
479           ∈ [POSSIBLE_EXPENSE_IDs → (Expense ∪ {NO_EXPENSE})]
480     ∧ replicas[rid].groups
481           ∈ [POSSIBLE_GROUP_IDs → (Group ∪ {NO_GROUP})]

483 Inv_Conservation_of_amount ≜
484   ∀ rid ∈ POSSIBLE_REPLICA_IDs :
485     ∀ eid ∈ POSSIBLE_EXPENSE_IDs :
486       replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE ⇒
487         LET e ≜ replicas[rid].recordedExpenses[eid]
488         IN  e.amount = SumFunction(e.shares)

490 Inv_ExpenseGroupExists ≜
491   ∀ rid ∈ POSSIBLE_REPLICA_IDs :
492     ∀ eid ∈ POSSIBLE_EXPENSE_IDs :
493       ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
494       ∧ replicas[rid].recordedExpenses[eid].deleted = FALSE      \* remove this?
495       ∧ replicas[rid].recordedExpenses[eid].group ≠ NO_GROUP
496       ⇒
497         replicas[rid].groups[replicas[rid].recordedExpenses[eid].group] ≠ NO_GROUP


500 Inv_GroupBalanceZero ≜
501   ∀ rid ∈ POSSIBLE_REPLICA_IDs :
502     ∀ gid ∈ POSSIBLE_GROUP_IDs :
503       replicas[rid].groups[gid] ≠ NO_GROUP ⇒
504         LET allUsers ≜
505                 include every user that was a member of the group at some part, as they always could accumulate negative bal
506               {u ∈ USERS : WasEverMember(replicas[rid].groups[gid].members[u])}
507             total ≜
508               SumFunction([u ∈ allUsers ↦ Balance(u, gid, replicas[rid])])
509         IN  total + replicas[rid].groups[gid].totalGifted = 0
```

```
Inv ≜
  ∧ TypeOK
  ∧ Inv_Conservation_of_amount
  ∧ Inv_ExpenseGroupExists
  ∧ Inv_GroupBalanceZero


  ----------------------------
  Liveness Helper
  ----------------------------
AllReplicasHaveAtLeastExpenseVersion(eid, version) ≜
    ∀ rid ∈ POSSIBLE_REPLICA_IDs :
          ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
          ∧ replicas[rid].recordedExpenses[eid].version ≥ version

AllReplicasHaveAtLeastGroupMemberCounter(gid, user, counter) ≜
    ∀ rid ∈ POSSIBLE_REPLICA_IDs :
        ∧ replicas[rid].groups[gid] ≠ NO_GROUP
        ∧ replicas[rid].groups[gid].members[user] ≥ counter

AllPositiveSharesAcknowledged(eid, replica) ≜
      replica.recordedExpenses[eid] = NO_EXPENSE
      ∨
    ∀ u ∈ USERS :
       replica.recordedExpenses[eid].shares[u] > 0
        ⇒ replica.recordedExpenses[eid].acknowledged_shares[u] = TRUE

  ----------------------------
  Liveness
  ----------------------------

  Expense version captures modifications and group membership
Liveness_ExpensePropagates ≜
    ∀ rid ∈ POSSIBLE_REPLICA_IDs :
       ∀ eid ∈ POSSIBLE_EXPENSE_IDs :
          □◇(replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
                ⇒ AllReplicasHaveAtLeastExpenseVersion(eid, replicas[rid].recordedExpenses[eid].version))
           ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
           ⇒ □◇(AllReplicasHaveAtLeastExpenseVersion(eid, replicas[rid].recordedExpenses[eid].version))

Liveness_GroupMemberShipPropagates ≜
    ∀ rid ∈ POSSIBLE_REPLICA_IDs :
       ∀ gid ∈ POSSIBLE_GROUP_IDs :
          ∀ user ∈ USERS :
             □◇(replicas[rid].groups[gid] ≠ NO_GROUP
                   ⇒ AllReplicasHaveAtLeastGroupMemberCounter(gid, user, replicas[rid].groups[gid].memb
```

```
559   Expenses in groups eventually resolve acknowledgment status
560   Either all shares are acknowledged or the expense is deleted/removed, or all members leave
561 Liveness_ExpenseSharesEventuallyAcknowledged ≜
562     ∀ rid ∈ POSSIBLE_REPLICA_IDs :
563       ∀ eid ∈ POSSIBLE_EXPENSE_IDs :
564         □◇(
565             LET exp ≜ replicas[rid].recordedExpenses[eid]
566             IN    ∧ exp ≠ NO_EXPENSE
567                   ∧ exp.group ≠ NO_GROUP
568                   ∧ ¬exp.deleted
569                   ⇒ ( ∨ AllPositiveSharesAcknowledged(eid, replicas[rid])
570                       ∨ exp.deleted
571                       ∨ exp.group = NO_GROUP
572                       ∨ ∀ u ∈ USERS : ¬IsMember(replicas[rid].groups[exp.group].members[u])))

      Safety Helper

578 NoDecreaseExpenseVersion ≜
579   ∀ rid ∈ POSSIBLE_REPLICA_IDs :
580   ∀ eid ∈ POSSIBLE_EXPENSE_IDs :
581     (replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE)
582       ⇒
583       ∧ replicas′[rid].recordedExpenses[eid] ≠ NO_EXPENSE
584       ∧ replicas′[rid].recordedExpenses[eid].version
585           ≥ replicas[rid].recordedExpenses[eid].version

587 NoDecreaseGroupMembersCounter ≜
588   ∀ rid ∈ POSSIBLE_REPLICA_IDs :
589   ∀ gid ∈ POSSIBLE_GROUP_IDs :
590   ∀ u ∈ USERS :
591     (replicas[rid].groups[gid] ≠ NO_GROUP)
592       ⇒
593       ∧ replicas′[rid].groups[gid] ≠ NO_GROUP
594       ∧ replicas′[rid].groups[gid].members[u]
595           ≥ replicas[rid].groups[gid].members[u]

597 NoDecreaseAcknowledgedSharesSameVersion ≜
598   ∀ rid ∈ POSSIBLE_REPLICA_IDs :
599   ∀ eid ∈ POSSIBLE_EXPENSE_IDs :
600   ∀ u ∈ USERS :
601     ∧ replicas[rid].recordedExpenses[eid] ≠ NO_EXPENSE
602       ∧ replicas′[rid].recordedExpenses[eid] ≠ NO_EXPENSE this always exists by NoDecreaseExpenseVersion property
603     ∧ replicas[rid].recordedExpenses[eid].version = replicas′[rid].recordedExpenses[eid].version
604     ∧ replicas[rid].recordedExpenses[eid].acknowledged_shares[u] = TRUE
605     ⇒
606       replicas′[rid].recordedExpenses[eid].acknowledged_shares[u] = TRUE
```

Safety

$Safety\_ExpenseVersionsNonDecreasing \triangleq$
$\quad \Box[NoDecreaseExpenseVersion]_{\{\langle replicas,\ actionCounter\rangle\}}$

$Safety\_GroupMembersCounterNonDecreasing \triangleq$
$\quad \Box[NoDecreaseGroupMembersCounter]_{\{\langle replicas,\ actionCounter\rangle\}}$

$Safety\_AcknowledgedSharesNonDecreasingForSameVersion \triangleq$
$\quad \Box[NoDecreaseAcknowledgedSharesSameVersion]_{\{\langle replicas,\ actionCounter\rangle\}}$

Specification

$Spec \triangleq Init \land \Box[Next]_{\langle replicas,\ actionCounter\rangle}$

$FairSpec \triangleq$
$\quad \land\ Spec$
$\quad \land\ \mathrm{WF}_{\langle replicas,\ actionCounter\rangle}(MergeReplicas)$
$\quad \land\ \mathrm{WF}_{\langle replicas,\ actionCounter\rangle}(AcknowledgeShare)$
$\quad \land\ \mathrm{WF}_{\langle replicas,\ actionCounter\rangle}(PayerAbsorbsLeftMemberShare)$
$\quad \land\ \mathrm{WF}_{\langle replicas,\ actionCounter\rangle}(RejoinToAcknowledge)$

---

* *LaTeX*: $\backslash usepackage[landscape,\ margin = 0.5in]\{geometry\}$
* Modification History
* Last modified *Wed Jan* 14 10:10:09 *CET* 2026 by *floyd*
* Created *Fri Oct* 24 11:14:17 *CEST* 2025 by *floyd*