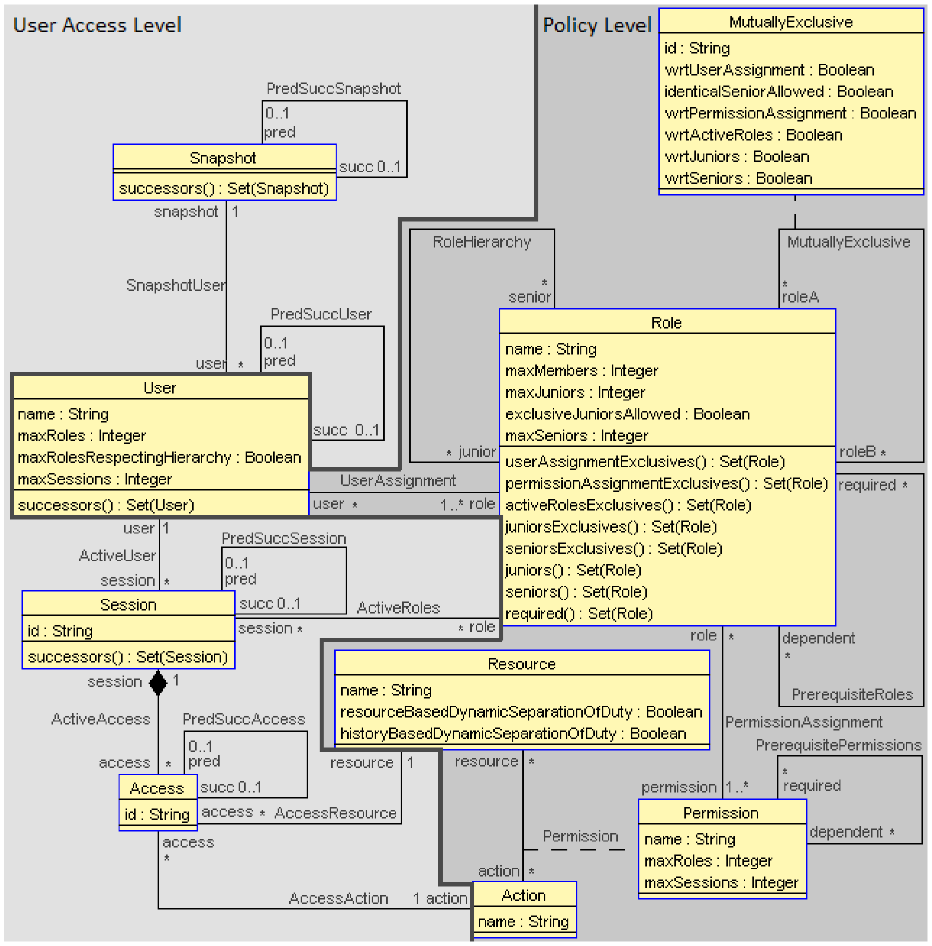
RBAC Metamodel: Sources and Validation Results

Mirco Kuhlmann[1], Karsten Sohr[2], and Martin Gogolla[1]

[1] University of Bremen, Computer Science Department
Database Systems Group, D-28334 Bremen, Germany
{mk|gogolla}@informatik.uni-bremen.de
[2] University of Bremen, Center for Computing Technologies
D-28334 Bremen, Germany
sohr@tzi.de

1 RBAC Metamodel Specification

1.1 UML



The RBAC metamodel is specified with the UML-based Specification Environment (USE) [1].

```
-----------------------------------------------------------------------
-----------------------------------------------------------------------

-- [1] Role-Based Access Control Models
-- Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman
-- \Literatur\RBAC\nocite\sandhu96.pdf

-- A:   user, role, permission, session
-- B:   permission to role assignment, user to role assignment

-- C:   active user (user : S -> U), active roles (roles : S -> 2^R)

-- D:   role hierarchies, junior roles, senior roles

-- E:   policy constraints (static)
-- E1:  mutually exclusive roles in terms of user assignment
-- E2:  mutually exclusive roles in terms of permission assignment
-- E3:  role: maximum number of members
-- E4:  user: maximum number of roles
-- E5:  permission: maximum number of roles
-- E6:  prerequisite roles
-- E7:  prerequisite permissions
-- E8:  mutually exclusive roles in terms of a session
-- E9:  user: maximum number of sessions active at the same time
-- E10: permission: maximum number of sessions activating the permission
--      at the same time
-- E11: role: maximum number of junior roles
-- E12: role: maximum number of senior roles
-- E13: roles: no common junior roles
-- E14: roles: no common senior roles
-- E15: mutually exclusive roles (in terms of user assignment)
--      may allow identical senior
-- E16: user: maximum number of roles respecting role hierarchy

----------------------------------------------------------------------

-- [2] Analyzing and Managing Role-Based Access Control Policies
-- Karsten Sohr, Michael Drouineaud, Gail-Joon Ahn, Senior Member, IEEE,
-- and Martin Gogolla
-- \Literatur\RBAC\nocite\AnalyzingAndManagingRBACPolicies.pdf

-- F:   actions and resources (operations and objects), resource access

-- G:   policy constraints
-- G1:  resource access only with respective permission
-- G2:  resource-based dynamic separation of duty
-- G3:  history-based dynamic separation of duty

----------------------------------------------------------------------

-- Z:   additional policy constraints (static)
-- Z1:  required roles must not be exclusive with respect to user assignment
-- Z2:  exclusion links must determine at least one particular exclusion
-- Z3:  roles must not be self-exclusive

-----------------------------------------------------------------------
-----------------------------------------------------------------------

-- SNAP: snapshot concepts

-- S:   snapshot constraints (dynamic)
-- S1:  snapshots establish one chain
-- S2:  successive users belong to successive snapshots
-- S3:  successive sessions related to successive users
-- S4:  successive accesses related to successive sessions
-- S5:  user names identify users over time
-- S6:  session ids identify sessions over time
-- S7:  access ids identify accesses over time

-----------------------------------------------------------------------
-----------------------------------------------------------------------

model RBAC

------------------- policy classes (static) ---------------------------
------------------- administrative view -------------------------------
------------------- no changes over time ------------------------------

class User ------------------------------------------------------ A [1]
attributes
  name : String ---------------------------------------------------- S5
  maxRoles : Integer ------------------------------------------- E4 [1]
  maxRolesRespectingHierarchy : Boolean ----------------------- E16 [1]
  maxSessions : Integer ---------------------------------------- E9 [1]
operations

-- For CD View:
--  successors() : Set(User) = Set{}

  successors() : Set(User) = if self.succ.isDefined then --------- SNAP
    successorsAux(Sequence{self.succ})->asSet() else Set{} endif
  successorsAux(users:Sequence(User)) : Sequence(User) =
    let successor = users->last().succ in
    if successor.isDefined() then
      successorsAux(users->append(successor))
    else users endif
end
```

```
class Role ------------------------------------------------------ A [1]
attributes
  name : String
  maxMembers : Integer ----------------------------------------- E3 [1]
  maxJuniors : Integer ---------------------------------------- E11 [1]
  exclusiveJuniorsAllowed : Boolean --------------------------- E15 [1]
  maxSeniors : Integer ---------------------------------------- E12 [1]
operations
  userAssignmentExclusives() : Set(Role) =
    mutuallyExclusive[roleA]->select(wrtUserAssignment).roleB
    ->union(
    mutuallyExclusive[roleB]->select(wrtUserAssignment).roleA)
    ->asSet()
  permissionAssignmentExclusives() : Set(Role) =
    mutuallyExclusive[roleA]->select(wrtPermissionAssignment).roleB
    ->union(
    mutuallyExclusive[roleB]->select(wrtPermissionAssignment).roleA)
    ->asSet()
  activeRolesExclusives() : Set(Role) =
    mutuallyExclusive[roleA]->select(wrtActiveRoles).roleB
    ->union(
    mutuallyExclusive[roleB]->select(wrtActiveRoles).roleA)
    ->asSet()
  juniorsExclusives() : Set(Role) =
    mutuallyExclusive[roleA]->select(wrtJuniors).roleB
    ->union(
    mutuallyExclusive[roleB]->select(wrtJuniors).roleA)
    ->asSet()
  seniorsExclusives() : Set(Role) =
    mutuallyExclusive[roleA]->select(wrtSeniors).roleB
    ->union(
    mutuallyExclusive[roleB]->select(wrtSeniors).roleA)
    ->asSet()

-- For CD View:
--  juniors() : Set(Role) = Set{}
--  seniors() : Set(Role) = Set{}
--  required() : Set(Role) = Set{}

  juniors() : Set(Role) = juniorsAux(self.junior) --------------- D [1]
  juniorsAux(roles:Set(Role)) : Set(Role) =
    let juniorRoles = roles.junior->asSet() in
    if roles->includesAll(juniorRoles) then roles
    else juniorsAux(roles->union(juniorRoles))
    endif
  seniors() : Set(Role) = seniorsAux(self.senior) --------------- D [1]
  seniorsAux(roles:Set(Role)) : Set(Role) =
    let seniorRoles = roles.senior->asSet() in
    if roles->includesAll(seniorRoles) then roles
    else seniorsAux(roles->union(seniorRoles))
    endif

  required() : Set(Role) = requiredAux(self.required) ---------- E6 [1]
  requiredAux(roles:Set(Role)) : Set(Role) =
    let requiredRoles = roles.required->asSet() in
    if roles->includesAll(requiredRoles) then roles
    else requiredAux(roles->union(requiredRoles))
    endif
end

associationclass Permission between --------------------- A [1] / F [2]
Action[*]
Resource[*]
attributes
  name : String
  maxRoles : Integer ------------------------------------------ E5 [1]
  maxSessions : Integer --------------------------------------- E10 [1]
end

class Action ---------------------------------------------------- F [2]
attributes
  name : String
end

class Resource -------------------------------------------------- F [2]
attributes
  name : String
  resourceBasedDynamicSeparationOfDuty : Boolean --------------- G2 [2]
  historyBasedDynamicSeparationOfDuty: Boolean ----------------- G3 [2]
end

------------------- user activity classes (dynamic) -------------------
------------------- user activity view --------------------------------
------------------- instances may change over time --------------------

class Session --------------------------------------------------- A [1]
attributes
  id : String ---------------------------------------------------- S6
operations
```

```
--  successors() : Set(Session) = Set{}
  successors() : Set(Session) = if self.succ.isDefined then ------ SNAP
    successorsAux(Sequence{self.succ})->asSet() else Set{} endif
  successorsAux(sessions:Sequence(Session)) : Sequence(Session) =
    let successor = sessions->last().succ in
    if successor.isDefined() then
      successorsAux(sessions->append(successor))
    else sessions endif
end

class Access ---------------------------------------------------- F [2]
attributes
  id : String ------------------------------------------------------- S7
end

------------------- policy associations (static) ----------------------
------------------- administrative view ------------------------------
------------------- no changes over time -----------------------------

association PermissionAssignment between ------------------------ B [1]
Permission[1..*]
Role[*]
end

association UserAssignment between ------------------------------ B [1]
User[*]
Role[1..*]
end

association RoleHierarchy between ------------------------------- D [1]
Role[*] role senior
Role[*] role junior
end

association PrerequisiteRoles between -------------------------- E6 [1]
Role[*] role required
Role[*] role dependent
end

association PrerequisitePermissions between -------------------- E7 [1]
Permission[*] role required
Permission[*] role dependent
end

associationclass MutuallyExclusive between ---------------------- E [1]
```

```
Role[*] role roleA
Role[*] role roleB
attributes
  id : String
  wrtUserAssignment : Boolean ---------------------------------- E1 [1]
  identicalSeniorAllowed : Boolean ----------------------------- E15 [1]
  wrtPermissionAssignment : Boolean ---------------------------- E2 [1]
  wrtActiveRoles : Boolean ------------------------------------- E8 [1]
  wrtJuniors : Boolean ---------------------------------------- E13 [1]
  wrtSeniors : Boolean ---------------------------------------- E14 [1]
end

------------------- user activity assocations (dynamic) ---------------
------------------- user activity view --------------------------------
------------------- instances may change over time --------------------

association ActiveUser between --------------------------------- C [1]
Session[*]
User[1]
end

association ActiveRoles between -------------------------------- C [1]
Session[*]
Role[*]
end

composition ActiveAccess between ------------------------------- F [2]
Session[1]
Access[*]
end

association AccessAction between ------------------------------- F [2]
Access[*]
Action[1]
end

association AccessResource between ----------------------------- F [2]
Access[*]
Resource[1]
end

------------------- snapshot concepts (dynamic) -----------------------
------------------- basis for dynamic user activity -------------------

class Snapshot --------------------------------------------------- SNAP
```

```
operations

-- For CD View:
--    successors() : Set(Snapshot) = Set{}

  successors() : Set(Snapshot) = successorsAux(Set{self.succ}) --- SNAP

  successorsAux(snapshots:Set(Snapshot)) : Set(Snapshot) =
    let succSnapshots = snapshots->collect(s |
        if s.succ.isDefined() then s.succ else s endif)->asSet() in
    if snapshots->includesAll(succSnapshots) then snapshots
    else successorsAux(snapshots->union(succSnapshots))
    endif
end

association PredSuccSnapshot between ----------------------------- SNAP
Snapshot[0..1] role pred
Snapshot[0..1] role succ
end

association PredSuccUser between --------------------------------- SNAP
User[0..1] role pred
User[0..1] role succ
end

association PredSuccSession between ------------------------------ SNAP
Session[0..1] role pred
Session[0..1] role succ
end

association PredSuccAccess between ------------------------------- SNAP
Access[0..1] role pred
Access[0..1] role succ
end

association SnapshotUser between --------------------------------- SNAP
Snapshot[1]
User[*]
end

-----------------------------------------------------------------------
-----------------------------------------------------------------------

constraints
```

```
------------------- policy constraints (static) -----------------------
------------------- administrative view -------------------------------
------------------- constraints independent from snapshots ------------

context u:User
  inv NoUserAssignedtoExclusiveRoles: -------------------------- E1 [1]
    u.role->forAll(r1, r2 |
      r1.userAssignmentExclusives()->excludes(r2))
  inv MaximumNumberOfRoles: -------------------------------- E4/E16 [1]
    let numberOfRoles =
      if u.maxRolesRespectingHierarchy.isDefined and
         u.maxRolesRespectingHierarchy then
           u.role->union(u.role.juniors())->asSet()->size()
      else u.role->size() endif
    in maxRoles.isDefined implies numberOfRoles <= maxRoles
  inv MaximumNumberOfSessions: ---------------------------------- E9 [1]
    u.maxSessions.isDefined implies u.session->size() <= u.maxSessions

context r:Role
  inv RoleHierarchyPartialOrder: -------------------------------- D [1]
    r.seniors()->excludes(r)
  inv MaximumNumberOfMembers: ----------------------------------- E3 [1]
    r.maxMembers.isDefined implies r.user->size() <= r.maxMembers
  inv RequiredRolesPresent: ------------------------------------- E6 [1]
    r.user->forAll(u | u.role->includesAll(r.required))
  inv RequiredRolesNotExclusive: ------------------------------------ Z1
    r.required()->excludesAll(r.userAssignmentExclusives())
  inv MaximumNumberOfJuniors: ---------------------------------- E11 [1]
    r.maxJuniors.isDefined implies r.junior->size() <= r.maxJuniors
  inv MaximumNumberOfSeniors: ---------------------------------- E12 [1]
    r.maxSeniors.isDefined implies r.senior->size() <= r.maxSeniors
  inv NoSharedJuniorsOfExclusiveRoles: ------------------------ E13 [1]
    r.juniors()->excludesAll(r.juniorsExclusives().juniors())
  inv NoSharedSeniorsOfExclusiveRoles: ------------------------ E14 [1]
    r.seniors()->excludesAll(r.seniorsExclusives().seniors())
    inv SeniorsWithExclusiveJuniors: ---------------------------- E15 [1]
    r.exclusiveJuniorsAllowed or
    r.juniors()->forAll(r1, r2 |
      r1.mutuallyExclusive[roleA]->union(r1.mutuallyExclusive[roleB])
        ->forAll(m | m.wrtUserAssignment.isDefined() and
                     m.wrtUserAssignment and m.roleB=r2 implies
                       m.identicalSeniorAllowed))

context p:Permission
  inv NoPermissionAssignedtoExclusiveRoles: -------------------- E2 [1]
```

```
    p.role->forAll(r1, r2 |
      r1.permissionAssignmentExclusives()->excludes(r2))
  inv MaximumNumberOfRoles: ----------------------------------- E5 [1]
    p.maxRoles.isDefined implies p.role->size() <= p.maxRoles
  inv RequiredPermissionsPresent: ------------------------------ E7 [1]
    p.role->forAll(r | r.permission->includesAll(p.required))

context s:Session
  inv ActiveRolesSubsetUserRoles: ------------------------------- C [1]
    let directUserRoles = s.user.role in
    let userRoles = directUserRoles->union(directUserRoles.juniors())
    in userRoles->includesAll(s.role)
  inv ActionsPermitted: --------------------------------------- G1 [2]
    s.access->forAll(a |
      let neededPermissions = a.action.permission
          ->select(p | p.resource = a.resource) in
      neededPermissions->notEmpty() and
      s.role.permission->union(s.role.juniors().permission)->asSet()
        ->includesAll(neededPermissions))

context m:MutuallyExclusive --------------------------------------- Z2
  inv DeterminationOfAtLeastOneExclusion:
    m.wrtUserAssignment or m.wrtPermissionAssignment or
    m.wrtActiveRoles or m.wrtJuniors or m.wrtSeniors
  inv NoSelfExclusion: --------------------------------------------- Z3
    m.roleA <> m.roleB

-------------------- policy constraints (dynamic) ----------------------
-------------------- administrative view ------------------------------
-------------------- constraints depending on snapshots ----------------

context p:Permission
  inv MaximumNumberOfSessions: ------------------------------- E10 [1]
    p.maxSessions.isDefined implies
      Snapshot.allInstances()->forAll(snap |
        p.role.session->asSet()->select(s |
          s.user.snapshot = snap)->size() <= p.maxSessions)

-- erroneous version
--  inv MaximumNumberOfSessions: ------------------------------- E10 [1]
--    p.maxSessions.isDefined implies
--      p.role.session->asSet()->size() <= p.maxSessions

context u:User
  inv ResourceBasedDynamicSeparationOfDuty: -------------------- G2 [2]
```

```
    Resource.allInstances()->forAll(r |
      r.resourceBasedDynamicSeparationOfDuty.isDefined and
      r.resourceBasedDynamicSeparationOfDuty implies
        r.access->select(a |
          u.successors()->including(u)->includes(a.session.user))
            .action->asSet()->size() <= 1)
  inv HistoryBasedDynamicSeparationOfDuty: --------------------- G3 [2]
    Resource.allInstances()->forAll(r |
      let availableActions = r.action->size() in
      r.historyBasedDynamicSeparationOfDuty.isDefined and
      r.historyBasedDynamicSeparationOfDuty and
      availableActions > 1 implies
        r.access->select(a |
          u.successors()->including(u)->includes(a.session.user))
            .action->asSet->size() < availableActions)

-- erroneous version
--  inv HistoryBasedDynamicSeparationOfDuty: --------------------- G3 [2]
--    Resource.allInstances()->forAll(r |
--      let availableActions = r.action->size() in
--      r.historyBasedDynamicSeparationOfDuty.isDefined and
--      r.historyBasedDynamicSeparationOfDuty implies
--        r.access->select(a |
--          u.successors()->including(u)->includes(a.session.user))
--            .action->asSet->size() < availableActions)

context s:Session
  inv NoExclusiveRolesActive: --------------------------------- E8 [1]
    let activeRoles = s.successors().role->union(s.role) in
    activeRoles->excludesAll(activeRoles.activeRolesExclusives())

------------------- snapshot constraints (dynamic) --------------------
------------------- basis for dynamic user activity -------------------

context snap:Snapshot
  inv ChainOfSnapshots: ------------------------------------------- S1
    let allSnapshots = Snapshot.allInstances in
    snap.successors()->excludes(snap) and
    allSnapshots->exists(s |
      s.successors()->includesAll(allSnapshots-Set{s}))

context u:User
  inv SuccUserInSuccSnapshot: ------------------------------------- S2
    u.succ.isDefined implies u.succ.snapshot = u.snapshot.succ
  inv UserNameIdentifies: ----------------------------------------- S5
```

```
    u.succ.isDefined implies u.succ.name = u.name

context s:Session
  inv SuccSessionRelatedToSuccUser: -------------------------------- S3
    s.succ.isDefined implies s.succ.user = s.user.succ
  inv SessionIdIdentifies: ------------------------------------------ S6
    s.succ.isDefined implies s.succ.id = s.id

context acc:Access
  inv SuccAccessRelatedToSuccSession: ------------------------------ S4
    acc.succ.isDefined implies acc.succ.session = acc.session.succ
  inv AccessIdIdentifies: ------------------------------------------ S7
    acc.succ.isDefined implies acc.succ.id = acc.id
```

1.2 Relational Logic

OCL2Kodkod translates the UML model into relational logic. The resulting constraints are shown below.

```
(User_name . univ) in User &&
(univ . User_name) in String &&
(all c: User |
  lone (c . User_name)) &&
(User_maxRoles . univ) in User &&
(univ . User_maxRoles) in Integer &&
(User_maxRolesRespectingHierarchy . univ) in User &&
(univ . User_maxRolesRespectingHierarchy) in Boolean &&
(all c: User |
  lone (c . User_maxRolesRespectingHierarchy)) &&
(User_maxSessions . univ) in User &&
(univ . User_maxSessions) in Integer &&
(Role_name . univ) in Role &&
(univ . Role_name) in String &&
(all c: Role |
  lone (c . Role_name)) &&
(Role_maxMembers . univ) in Role &&
(univ . Role_maxMembers) in Integer &&
(Role_maxJuniors . univ) in Role &&
(univ . Role_maxJuniors) in Integer &&
(Role_exclusiveJuniorsAllowed . univ) in Role &&
(univ . Role_exclusiveJuniorsAllowed) in Boolean &&
(all c: Role |
  lone (c . Role_exclusiveJuniorsAllowed)) &&
(Role_maxSeniors . univ) in Role &&
(univ . Role_maxSeniors) in Integer &&
((Permission_assoc . univ) . univ) in Permission &&
((univ . Permission_assoc) . univ) in Action &&
(univ . (univ . Permission_assoc)) in Resource &&
(all c2: Action, c3: Resource |
  lone ((Permission_assoc . c3) . c2)) &&
(all c1: Permission |
  one (c1 . Permission_assoc)) &&
(Permission_name . univ) in Permission &&
(univ . Permission_name) in String &&
(all c: Permission |
  lone (c . Permission_name)) &&
(Permission_maxRoles . univ) in Permission &&
(univ . Permission_maxRoles) in Integer &&
(Permission_maxSessions . univ) in Permission &&
(univ . Permission_maxSessions) in Integer &&
(Action_name . univ) in Action &&
(univ . Action_name) in String &&
(all c: Action |
  lone (c . Action_name)) &&
(Resource_name . univ) in Resource &&
(univ . Resource_name) in String &&
(all c: Resource |
  lone (c . Resource_name)) &&
(Resource_resourceBasedDynamicSeparationOfDuty . univ) in Resource &&
(univ . Resource_resourceBasedDynamicSeparationOfDuty) in Boolean &&
(all c: Resource |
  lone (c . Resource_resourceBasedDynamicSeparationOfDuty)) &&
(Resource_historyBasedDynamicSeparationOfDuty . univ) in Resource &&
(univ . Resource_historyBasedDynamicSeparationOfDuty) in Boolean &&
(all c: Resource |
  lone (c . Resource_historyBasedDynamicSeparationOfDuty)) &&
(Session_id . univ) in Session &&
(univ . Session_id) in String &&
(all c: Session |
  lone (c . Session_id)) &&
(Access_id . univ) in Access &&
(univ . Access_id) in String &&
(all c: Access |
  lone (c . Access_id)) &&
(PermissionAssignment . univ) in Permission &&
(univ . PermissionAssignment) in Role &&
(all c2: Role |
  some (PermissionAssignment . c2)) &&
(UserAssignment . univ) in User &&
(univ . UserAssignment) in Role &&
(all c1: User |
  some (c1 . UserAssignment)) &&
(RoleHierarchy . univ) in Role &&
(univ . RoleHierarchy) in Role &&
(PrerequisiteRoles . univ) in Role &&
(univ . PrerequisiteRoles) in Role &&
(PrerequisitePermissions . univ) in Permission &&
(univ . PrerequisitePermissions) in Permission &&
((MutuallyExclusive_assoc . univ) . univ) in MutuallyExclusive &&
((univ . MutuallyExclusive_assoc) . univ) in Role &&
(univ . (univ . MutuallyExclusive_assoc)) in Role &&
(all c2: Role, c3: Role |
  lone ((MutuallyExclusive_assoc . c3) . c2)) &&
(all c1: MutuallyExclusive |
  one (c1 . MutuallyExclusive_assoc)) &&
(MutuallyExclusive_id . univ) in MutuallyExclusive &&
(univ . MutuallyExclusive_id) in Boolean &&
(all c: MutuallyExclusive |
  lone (c . MutuallyExclusive_id)) &&
(MutuallyExclusive_wrtUserAssignment . univ) in MutuallyExclusive &&
(univ . MutuallyExclusive_wrtUserAssignment) in Boolean &&
(all c: MutuallyExclusive |
  lone (c . MutuallyExclusive_wrtUserAssignment)) &&
(MutuallyExclusive_identicalSeniorAllowed . univ) in MutuallyExclusive &&
(univ . MutuallyExclusive_identicalSeniorAllowed) in Boolean &&
(all c: MutuallyExclusive |
  lone (c . MutuallyExclusive_identicalSeniorAllowed)) &&
(MutuallyExclusive_wrtPermissionAssignment . univ) in MutuallyExclusive &&
(univ . MutuallyExclusive_wrtPermissionAssignment) in Boolean &&
(all c: MutuallyExclusive |
  lone (c . MutuallyExclusive_wrtPermissionAssignment)) &&
(MutuallyExclusive_wrtActiveRoles . univ) in MutuallyExclusive &&
(univ . MutuallyExclusive_wrtActiveRoles) in Boolean &&
(all c: MutuallyExclusive |
  lone (c . MutuallyExclusive_wrtActiveRoles)) &&
(MutuallyExclusive_wrtJuniors . univ) in MutuallyExclusive &&
(univ . MutuallyExclusive_wrtJuniors) in Boolean &&
(all c: MutuallyExclusive |
  lone (c . MutuallyExclusive_wrtJuniors)) &&
(MutuallyExclusive_wrtJuniors . univ) in MutuallyExclusive &&
(univ . MutuallyExclusive_wrtJuniors) in Boolean &&
(all c: MutuallyExclusive |
  lone (c . MutuallyExclusive_wrtJuniors)) &&
(MutuallyExclusive_wrtSeniors . univ) in MutuallyExclusive &&
(univ . MutuallyExclusive_wrtSeniors) in Boolean &&
(all c: MutuallyExclusive |
  lone (c . MutuallyExclusive_wrtSeniors)) &&
(ActiveUser . univ) in Session &&
(univ . ActiveUser) in User &&
(all c1: Session |
  one (c1 . ActiveUser)) &&
(ActiveRoles . univ) in Session &&
(univ . ActiveRoles) in Role &&
(ActiveAccess . univ) in Session &&
(univ . ActiveAccess) in Access &&
(all c2: Access |
  one (ActiveAccess . c2)) &&
(AccessAction . univ) in Access &&
(univ . AccessAction) in Action &&
(all c1: Access |
  one (c1 . AccessAction)) &&
(AccessResource . univ) in Access &&
(univ . AccessResource) in Resource &&
(all c1: Access |
  one (c1 . AccessResource)) &&
(PredSuccSnapshot . univ) in Snapshot &&
(univ . PredSuccSnapshot) in Snapshot &&
(all c2: Snapshot |
  lone (PredSuccSnapshot . c2)) &&
(all c1: Snapshot |
  lone (c1 . PredSuccSnapshot)) &&
(PredSuccUser . univ) in User &&
(univ . PredSuccUser) in User &&
(all c2: User |
  lone (PredSuccUser . c2)) &&
(all c1: User |
  lone (c1 . PredSuccUser)) &&
(PredSuccSession . univ) in Session &&
(univ . PredSuccSession) in Session &&
(all c2: Session |
  lone (PredSuccSession . c2)) &&
(all c1: Session |
  lone (c1 . PredSuccSession)) &&
(PredSuccAccess . univ) in Access &&
(univ . PredSuccAccess) in Access &&
(all c2: Access |
  lone (PredSuccAccess . c2)) &&
(all c1: Access |
  lone (c1 . PredSuccAccess)) &&
(SnapshotUser . univ) in Snapshot &&
(univ . SnapshotUser) in User &&
(all c2: User |
  one (SnapshotUser . c2)) &&
(all u: User |
  all r1: u . UserAssignment, r2: u . UserAssignment |
   !(r2 in ((univ . ({x1: (MutuallyExclusive_assoc . univ) . r1 | (x1 .
     MutuallyExclusive_wrtUserAssignment) = Boolean_true} .
     MutuallyExclusive_assoc)) + (({x2: (MutuallyExclusive_assoc . r1) . univ |
     (x2 . MutuallyExclusive_wrtUserAssignment) = Boolean_true} .
     MutuallyExclusive_assoc) . univ)))) &&
(all u: User |
  some (u . User_maxRoles) =>
  ((some (u . User_maxRolesRespectingHierarchy) &&
  (u . User_maxRolesRespectingHierarchy) = Boolean_true) =>
   #((u . UserAssignment) + ((u . UserAssignment) . ^RoleHierarchy)) else
   #(u . UserAssignment)) <= #(u . User_maxRoles)) &&
(all u: User |
  some (u . User_maxSessions) =>
  #(ActiveUser . u) <= #(u . User_maxSessions)) &&
(all r: Role |
  !(r in (^RoleHierarchy . r))) &&
(all r: Role |
  some (r . Role_maxMembers) =>
  #(UserAssignment . r) <= #(r . Role_maxMembers)) &&
(all r: Role |
  all u: UserAssignment . r |
   (PrerequisiteRoles . r) in (u . UserAssignment)) &&
(all r: Role |
  no ((^PrerequisiteRoles . r) & ((univ . ({x1: (MutuallyExclusive_assoc . univ) .
  r | (x1 . MutuallyExclusive_wrtUserAssignment) = Boolean_true} .
  MutuallyExclusive_assoc)) + (({x2: (MutuallyExclusive_assoc . r) . univ | (x2 .
  MutuallyExclusive_wrtUserAssignment) = Boolean_true} . MutuallyExclusive_assoc
  ) . univ)))) &&
(all r: Role |
  some (r . Role_maxJuniors) =>
  #(r . RoleHierarchy) <= #(r . Role_maxJuniors)) &&
(all r: Role |
  some (r . Role_maxSeniors) =>
  #(RoleHierarchy . r) <= #(r . Role_maxSeniors)) &&
(all r: Role |
  no ((r . ^RoleHierarchy) & (((univ . ({x1: (MutuallyExclusive_assoc . univ) .
  r | (x1 . MutuallyExclusive_wrtJuniors) = Boolean_true} .
  MutuallyExclusive_assoc)) + (({x2: (MutuallyExclusive_assoc . r) . univ | (x2 .
  MutuallyExclusive_wrtJuniors) = Boolean_true} . MutuallyExclusive_assoc) .
  univ)) . ^RoleHierarchy))) &&
(all r: Role |
  no ((^RoleHierarchy . r) & (^RoleHierarchy . ((univ . ({x1: (
  MutuallyExclusive_assoc . univ) . r | (x1 . MutuallyExclusive_wrtSeniors) =
  Boolean_true} . MutuallyExclusive_assoc)) + (({x2: (MutuallyExclusive_assoc .
  r) . univ | (x2 . MutuallyExclusive_wrtSeniors) = Boolean_true} .
  MutuallyExclusive_assoc) . univ))))) &&
(all r: Role |
  (r . Role_exclusiveJuniorsAllowed) = Boolean_true ||
  (all r1: r . ^RoleHierarchy, r2: r . ^RoleHierarchy |
    all m: ((MutuallyExclusive_assoc . univ) . r1) + ((MutuallyExclusive_assoc .
    r1) . univ) |
     !(some (m . MutuallyExclusive_wrtUserAssignment) &&
       (m . MutuallyExclusive_wrtUserAssignment) = Boolean_true &&
       (univ . (m . MutuallyExclusive_assoc)) = r2) ||
     (m . MutuallyExclusive_identicalSeniorAllowed) = Boolean_true)) &&
(all p: Permission |
  all r1: p . PermissionAssignment, r2: p . PermissionAssignment |
   !(r2 in ((univ . ({x1: (MutuallyExclusive_assoc . univ) . r1 | (x1 .
     MutuallyExclusive_wrtPermissionAssignment) = Boolean_true} .
     MutuallyExclusive_assoc)) + (({x2: (MutuallyExclusive_assoc . r1) . univ |
     (x2 . MutuallyExclusive_wrtPermissionAssignment) = Boolean_true} .
     MutuallyExclusive_assoc) . univ)))) &&
(all p: Permission |
  some (p . Permission_maxRoles) =>
  #(p . PermissionAssignment) <= #(p . Permission_maxRoles)) &&
(all p: Permission |
  all r: p . PermissionAssignment |
   (PrerequisitePermissions . p) in (PermissionAssignment . r)) &&
(all s: Session |
  (s . ActiveRoles) in (((s . ActiveUser) . UserAssignment) + (((s . ActiveUser) .
  UserAssignment) . ^RoleHierarchy))) &&
(all s: Session |
  no ((s . ActiveRoles) & ((univ . ({x1: (MutuallyExclusive_assoc . univ) . (s .
  ActiveRoles) | (x1 . MutuallyExclusive_wrtActiveRoles) = Boolean_true} .
  MutuallyExclusive_assoc)) + (({x2: (MutuallyExclusive_assoc . (s . ActiveRoles
  )) . univ | (x2 . MutuallyExclusive_wrtActiveRoles) = Boolean_true} .
  MutuallyExclusive_assoc) . univ)))) &&
(all s: Session |
  all a: s . ActiveAccess |
   some {p: (Permission_assoc . univ) . (a . AccessAction) | (univ . (p .
   Permission_assoc)) = (a . AccessResource)} &&
   {p: (Permission_assoc . univ) . (a . AccessAction) | (univ . (p .
   Permission_assoc)) = (a . AccessResource)} in ((PermissionAssignment . (s .
   ActiveRoles)) + (PermissionAssignment . ((s . ActiveRoles) . ^RoleHierarchy))
   )) &&
(all m: MutuallyExclusive |
  (m . MutuallyExclusive_wrtUserAssignment) = Boolean_true ||
  (m . MutuallyExclusive_wrtPermissionAssignment) = Boolean_true ||
  (m . MutuallyExclusive_wrtActiveRoles) = Boolean_true ||
  (m . MutuallyExclusive_wrtJuniors) = Boolean_true ||
  (m . MutuallyExclusive_wrtSeniors) = Boolean_true) &&
(all m: MutuallyExclusive |
  !(((m . MutuallyExclusive_assoc) . univ) = (univ . (m .
    MutuallyExclusive_assoc)))) &&
(all p: Permission |
  some (p . Permission_maxSessions) =>
  (all snap: Snapshot |
    #({s: ActiveRoles . (p . PermissionAssignment) | (SnapshotUser . (s .
    ActiveUser)) = snap}) <= #(p . Permission_maxSessions))) &&
(all u: User |
  all r: Resource |
   (some (r . Resource_resourceBasedDynamicSeparationOfDuty) &&
    (r . Resource_resourceBasedDynamicSeparationOfDuty) = Boolean_true) =>
   lone ({a: AccessResource . r | ((ActiveAccess . a) . ActiveUser) in ((u . ^
   PredSuccUser) + u)} . AccessAction)) &&
(all u: User |
  all r: Resource |
   (some (r . Resource_historyBasedDynamicSeparationOfDuty) &&
    (r . Resource_historyBasedDynamicSeparationOfDuty) = Boolean_true &&
    #((univ . Permission_assoc) . r) > 1) =>
   #({a: AccessResource . r | ((ActiveAccess . a) . ActiveUser) in ((u . ^
   PredSuccUser) + u)} . AccessAction) < #((univ . Permission_assoc) . r)) &&
(all snap: Snapshot |
  !(snap in (snap . ^PredSuccSnapshot)) &&
  (some s: Snapshot |
    (Snapshot - s) in (s . ^PredSuccSnapshot))) &&
(all u: User |
  some (u . PredSuccUser) =>
  (SnapshotUser . (u . PredSuccUser)) = ((SnapshotUser . u) . PredSuccSnapshot)) &&
(all u: User |
  some (u . PredSuccUser) =>
  ((u . PredSuccUser) . User_name) = (u . User_name)) &&
(all s: Session |
  some (s . PredSuccSession) =>
  ((s . PredSuccSession) . ActiveUser) = ((s . ActiveUser) . PredSuccUser)) &&
(all s: Session |
  some (s . PredSuccSession) =>
  ((s . PredSuccSession) . Session_id) = (s . Session_id)) &&
(all acc: Access |
  some (acc . PredSuccAccess) =>
  (ActiveAccess . (acc . PredSuccAccess)) = ((ActiveAccess . acc) .
  PredSuccSession)) &&
(all acc: Access |
  some (acc . PredSuccAccess) =>
  ((acc . PredSuccAccess) . Access_id) = (acc . Access_id)) &&
```

2 Exemplary Object Diagram of Fig. 1

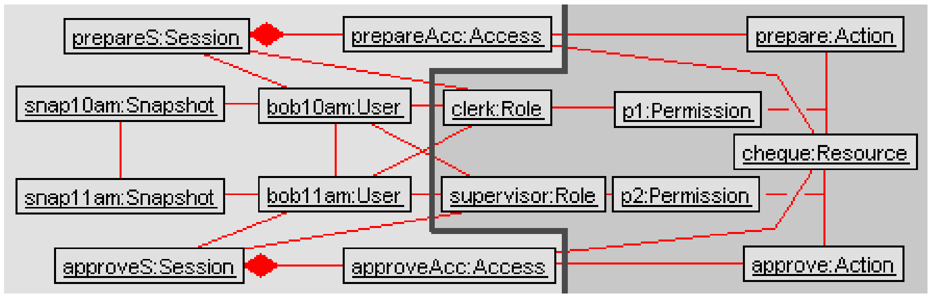


Diagram created with the following commands in USE (manually):

```
!create snap10am,snap11am : Snapshot
!create bob10am,bob11am : User
!create prepareS,approveS : Session
!create prepareAcc,approveAcc : Access
!create cheque : Resource
!create prepare,approve : Action
!create clerk,supervisor : Role
!create p1:Permission between (prepare,cheque)
!create p2:Permission between (approve,cheque)
!insert (snap10am,snap11am) into PredSuccSnapshot
!insert (bob10am,bob11am) into PredSuccUser
!insert (snap10am,bob10am) into SnapshotUser
!insert (snap11am,bob11am) into SnapshotUser
!insert (prepareS,bob10am) into ActiveUser
!insert (prepareS,prepareAcc) into ActiveAccess
!insert (prepareS,clerk) into ActiveRoles
!insert (approveS,approveAcc) into ActiveAccess
!insert (approveS,bob11am) into ActiveUser
!insert (approveS,supervisor) into ActiveRoles
!insert (p2,supervisor) into PermissionAssignment
!insert (p1,clerk) into PermissionAssignment
!insert (prepareAcc,prepare) into AccessAction
!insert (prepareAcc,cheque) into AccessResource
!insert (approveAcc,approve) into AccessAction
!insert (approveAcc,cheque) into AccessResource
!insert (bob10am,clerk) into UserAssignment
!insert (bob10am,supervisor) into UserAssignment
```

```
!insert (bob11am,clerk) into UserAssignment
!insert (bob11am,supervisor) into UserAssignment
```

3 Analyzing the RBAC Metamodel

3.1 Consistency

Each authorization constraint must be activated at least once. See the corresponding constraints in relational logic below.

```
some User_maxRoles &&
some User_maxSessions &&
some Role_maxMembers &&
some Role_maxJuniors &&
some Role_maxSeniors &&
some PrerequisiteRoles &&
(univ . MutuallyExclusive_wrtUserAssignment) = Boolean_true &&
(univ . MutuallyExclusive_wrtPermissionAssignment) = Boolean_true &&
(univ . MutuallyExclusive_wrtActiveRoles) = Boolean_true &&
(univ . MutuallyExclusive_wrtJuniors) = Boolean_true &&
(univ . MutuallyExclusive_wrtSeniors) = Boolean_true &&
some Permission_maxRoles &&
some Permission_maxSessions &&
some PrerequisitePermissions &&
(univ . Resource_resourceBasedDynamicSeparationOfDuty) = Boolean_true &&
(univ . Resource_historyBasedDynamicSeparationOfDuty) = Boolean_true
```

Internal Kodkod result:

```
---INSTANCE---
relations: {
Integer=[[integer1], [integer2]],
String=[[string1]],
Boolean=[[Boolean_true], [Boolean_false]],
Boolean_true=[[Boolean_true]],
User=[[user1], [user2]], User_name=[],
User_maxRoles=[[user2, integer1], [user2, integer2]],
User_maxRolesRespectingHierarchy=[],
User_maxSessions=[[user2, integer2]],
Role=[[role1], [role2]],
Role_name=[],
Role_maxMembers=[[role2, integer1]],
Role_maxJuniors=[[role1, integer1]],
Role_exclusiveJuniorsAllowed=[[role1, Boolean_true]],
Role_maxSeniors=[[role2, integer1], [role2, integer2]],
Permission=[[permission1], [permission2]],
Permission_name=[],
Permission_maxRoles=[[permission1, integer1]],
Permission_maxSessions=[[permission2, integer1]],
Action=[[action1], [action2]],
Action_name=[],
Resource=[[resource1], [resource2]],
Resource_name=[],
Resource_resourceBasedDynamicSeparationOfDuty=[resource2, Boolean_true],
Resource_historyBasedDynamicSeparationOfDuty=[[resource1, Boolean_true]],
Session=[[session1]],
Session_id=[],
Access=[[access1]],
Access_id=[],
Permission_assoc=[[permission1, action2, resource1], [permission2, action2, resource2]],
PermissionAssignment=[[permission1, role1], [permission2, role2]],
UserAssignment=[[user1, role1], [user2, role2]],
RoleHierarchy=[[role2, role1]],
PrerequisiteRoles=[[role2, role2]],
PrerequisitePermissions=[[permission1, permission1]],
MutuallyExclusive=[[mutuallyExclusive1], [mutuallyExclusive2]],
MutuallyExclusive_assoc=[[mutuallyExclusive1, role1, role2],
                         [mutuallyExclusive2, role2, role1]],
MutuallyExclusive_id=[],
MutuallyExclusive_wrtUserAssignment=[[mutuallyExclusive1, Boolean_true]],
MutuallyExclusive_identicalSeniorAllowed=[[mutuallyExclusive1, Boolean_true],
                                          [mutuallyExclusive2, Boolean_true]],
MutuallyExclusive_wrtPermissionAssignment=[[mutuallyExclusive2, Boolean_true]],
MutuallyExclusive_wrtActiveRoles=[[mutuallyExclusive1, Boolean_true]],
MutuallyExclusive_wrtJuniors=[[mutuallyExclusive1, Boolean_true]],
MutuallyExclusive_wrtSeniors=[[mutuallyExclusive1, Boolean_true]],
ActiveUser=[[session1, user2]],
ActiveRoles=[[session1, role1]],
ActiveAccess=[[session1, access1]],
AccessAction=[[access1, action2]],
AccessResource=[[access1, resource1]],
Snapshot=[[snapshot1]],
PredSuccSnapshot=[],
PredSuccUser=[],
PredSuccSession=[],
PredSuccAccess=[],
SnapshotUser=[[snapshot1, user1], [snapshot1, user2]],
Boolean_false=[[Boolean_false]]}
ints: []

---STATS---
```

```
p cnf 1509 2785
primary variables: 187
translation time: 94 ms
solving time: 16 ms
```

OCL2Kodkod Result (translation of the internal result into an object diagram, i.e., a sequence of USE commands):

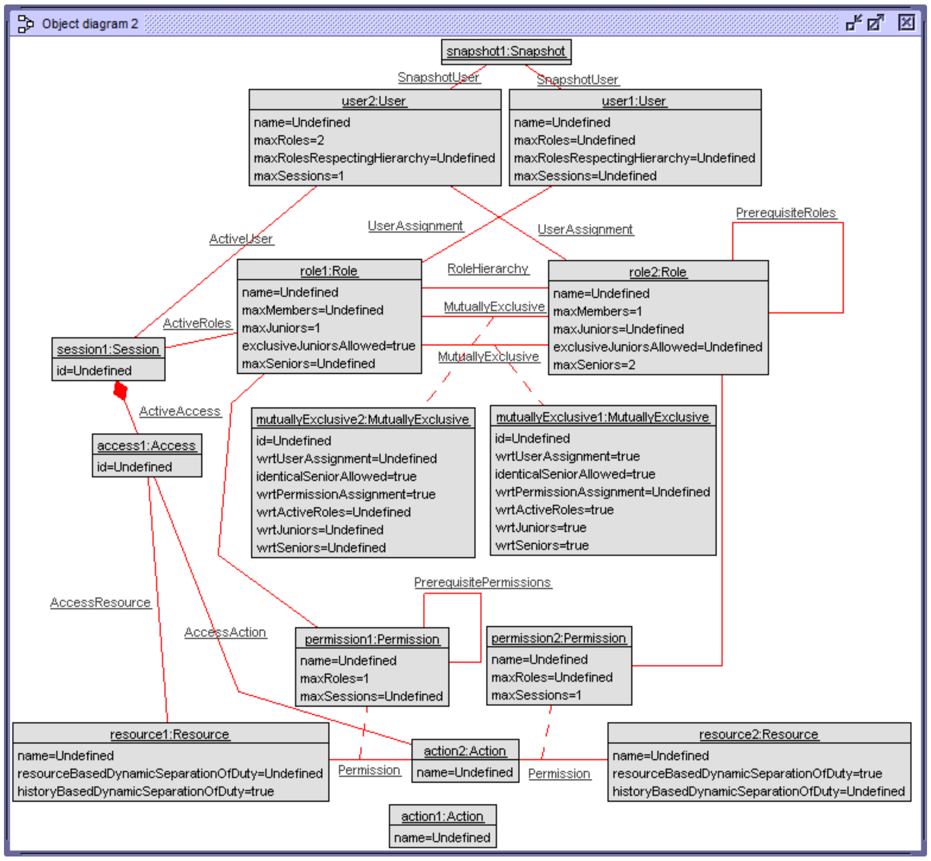


Diagram created with the following commands in USE:

```
reset
!create user1:User
!create user2:User
!create session1:Session
!create role1:Role
!create role2:Role
!create action1:Action
!create action2:Action
!create resource1:Resource
!create resource2:Resource
```

```
!create access1:Access
!create snapshot1:Snapshot
!create permission1:Permission between(action2, resource1)
!create permission2:Permission between(action2, resource2)
!create mutuallyExclusive1:MutuallyExclusive between(role1, role2)
!create mutuallyExclusive2:MutuallyExclusive between(role2, role1)
!set mutuallyExclusive1.wrtJuniors := true
!set mutuallyExclusive2.identicalSeniorAllowed := true
!set role2.maxMembers := 1
!set mutuallyExclusive1.identicalSeniorAllowed := true
!set role2.maxSeniors := 2
!set user2.maxSessions := 1
!set permission1.maxRoles := 1
!set mutuallyExclusive1.wrtActiveRoles := true
!set permission2.maxSessions := 1
!set role1.maxJuniors := 1
!set user2.maxRoles := 2
!set mutuallyExclusive2.wrtPermissionAssignment := true
!set resource1.historyBasedDynamicSeparationOfDuty := true
!set resource2.resourceBasedDynamicSeparationOfDuty := true
!set role1.exclusiveJuniorsAllowed := true
!set mutuallyExclusive1.wrtSeniors := true
!set mutuallyExclusive1.wrtUserAssignment := true
!insert (permission1,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (user1,  role1) into UserAssignment
!insert (user2,  role2) into UserAssignment
!insert (session1,  user2) into ActiveUser
!insert (session1,  role1) into ActiveRoles
!insert (role2,  role1) into RoleHierarchy
!insert (role2,  role2) into PrerequisiteRoles
!insert (permission1,  permission1) into PrerequisitePermissions
!insert (session1,  access1) into ActiveAccess
!insert (access1,  action2) into AccessAction
!insert (access1,  resource1) into AccessResource
!insert (snapshot1,  user1) into SnapshotUser
!insert (snapshot1,  user2) into SnapshotUser
```

3.2 Independence

For each invariant we present the object diagram (in form of a USE command sequence) returned by OCL2Kodkod and the respective USE validation result which shows that only the considered invariant is violated.

Access::AccessIdIdentifies

```
Access_AccessIdIdentifies Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create user2:User
!create session1:Session
!create session2:Session
!create role1:Role
!create role2:Role
!create action1:Action
!create action2:Action
!create resource2:Resource
!create access1:Access
!create access2:Access
!create snapshot1:Snapshot
!create snapshot2:Snapshot
!create permission2:Permission between(action2, resource2)
!create mutuallyExclusive1:MutuallyExclusive between(role2, role1)
!create mutuallyExclusive2:MutuallyExclusive between(role1, role2)
!set action1.name := 'string1'
!set user2.maxRolesRespectingHierarchy := false
!set mutuallyExclusive2.identicalSeniorAllowed := true
!set access1.id := 'string2'
!set action2.name := 'string2'
!set mutuallyExclusive1.identicalSeniorAllowed := false
!set mutuallyExclusive2.wrtUserAssignment := false
!set access2.id := 'string1'
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set mutuallyExclusive1.wrtActiveRoles := true
!set role1.maxJuniors := 1
!set resource2.name := 'string1'
!set mutuallyExclusive2.wrtSeniors := true
!set role1.exclusiveJuniorsAllowed := true
!set user1.maxRolesRespectingHierarchy := true
!set role2.name := 'string2'
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (user1,  role1) into UserAssignment
!insert (user2,  role1) into UserAssignment
!insert (session1,  user2) into ActiveUser
!insert (session2,  user1) into ActiveUser
!insert (session1,  role2) into ActiveRoles
!insert (session2,  role2) into ActiveRoles
!insert (role1,  role2) into RoleHierarchy
```

```
!insert (session1,  access2) into ActiveAccess
!insert (session2,  access1) into ActiveAccess
!insert (access1,  action2) into AccessAction
!insert (access2,  action2) into AccessAction
!insert (access1,  resource2) into AccessResource
!insert (access2,  resource2) into AccessResource
!insert (snapshot2,  snapshot1) into PredSuccSnapshot
!insert (user2,  user1) into PredSuccUser
!insert (session1,  session2) into PredSuccSession
!insert (access2,  access1) into PredSuccAccess
!insert (snapshot1,  user1) into SnapshotUser
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': FAILED.
  -> false : Boolean
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) `Session::SessionIdIdentifies': OK.
checking invariant (22) `Session::SuccSessionRelatedToSuccUser': OK.
checking invariant (23) `Snapshot::ChainOfSnapshots': OK.
checking invariant (24) `User::HistoryBasedDynamicSeparationOfDuty': OK.
checking invariant (25) `User::MaximumNumberOfRoles': OK.
checking invariant (26) `User::MaximumNumberOfSessions': OK.
checking invariant (27) `User::NoUserAssignedtoExclusiveRoles': OK.
checking invariant (28) `User::ResourceBasedDynamicSeparationOfDuty': OK.
checking invariant (29) `User::SuccUserInSuccSnapshot': OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.032s, 1 failure.
```

Access::SuccAccessRelatedToSuccSession

```
Access_SuccAccessRelatedToSuccSession Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create session1:Session
!create role1:Role
!create action1:Action
!create resource1:Resource
!create access1:Access
!create snapshot1:Snapshot
!create permission1:Permission between(action1, resource1)
!insert (permission1,  role1) into PermissionAssignment
!insert (user1,  role1) into UserAssignment
!insert (session1,  user1) into ActiveUser
!insert (session1,  role1) into ActiveRoles
!insert (session1,  access1) into ActiveAccess
!insert (access1,  action1) into AccessAction
!insert (access1,  resource1) into AccessResource
!insert (access1,  access1) into PredSuccAccess
!insert (snapshot1,  user1) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: FAILED.
  -> false : Boolean
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.016s, 1 failure.
```

MutuallyExclusive::DeterminationOfAtLeastOneExclusion

```
MutuallyExclusive_DeterminationOfAtLeastOneExclusion Independence
-----------------------------------------------------------------------

reset
!create user2:User
!create role1:Role
!create role2:Role
!create action2:Action
!create resource2:Resource
!create snapshot2:Snapshot
!create permission2:Permission between(action2, resource2)
!create mutuallyExclusive2:MutuallyExclusive between(role1, role2)
!set mutuallyExclusive2.identicalSeniorAllowed := false
!set role2.maxMembers := 1
!set mutuallyExclusive2.wrtUserAssignment := false
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set role2.maxJuniors := 1
!set mutuallyExclusive2.wrtJuniors := false
!set mutuallyExclusive2.wrtActiveRoles := false
!set permission2.maxSessions := 1
!set role1.maxJuniors := 1
!set mutuallyExclusive2.wrtPermissionAssignment := false
!set mutuallyExclusive2.wrtSeniors := false
!set role1.exclusiveJuniorsAllowed := true
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (user2,  role2) into UserAssignment
```

```
!insert (role1,  role2) into RoleHierarchy
!insert (role2,  role1) into PrerequisiteRoles
!insert (role2,  role2) into PrerequisiteRoles
!insert (permission2,  permission2) into PrerequisitePermissions
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: FAILED.
  -> false : Boolean
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

MutuallyExclusive::NoSelfExclusion

```
MutuallyExclusive_NoSelfExclusion Independence
-----------------------------------------------------------------------
```

```
reset
!create role1:Role
!create action1:Action
!create resource1:Resource
!create permission1:Permission between(action1, resource1)
!create mutuallyExclusive1:MutuallyExclusive between(role1, role1)
!set mutuallyExclusive1.wrtUserAssignment := true
!insert (permission1,  role1) into PermissionAssignment

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': FAILED.
  -> false : Boolean
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) `Session::SessionIdIdentifies': OK.
checking invariant (22) `Session::SuccSessionRelatedToSuccUser': OK.
checking invariant (23) `Snapshot::ChainOfSnapshots': OK.
checking invariant (24) `User::HistoryBasedDynamicSeparationOfDuty': OK.
checking invariant (25) `User::MaximumNumberOfRoles': OK.
checking invariant (26) `User::MaximumNumberOfSessions': OK.
checking invariant (27) `User::NoUserAssignedtoExclusiveRoles': OK.
checking invariant (28) `User::ResourceBasedDynamicSeparationOfDuty': OK.
checking invariant (29) `User::SuccUserInSuccSnapshot': OK.
checking invariant (30) `User::UserNameIdentifies': OK.
checked 30 invariants in 0.016s, 1 failure.
```

Permission::MaximumNumberOfRoles

```
Permission_MaximumNumberOfRoles Independence
-----------------------------------------------------------------------

reset
!create role1:Role
!create role2:Role
!create action1:Action
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create permission1:Permission between(action2, resource1)
!create permission2:Permission between(action1, resource2)
!set action2.name := 'string2'
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set resource1.name := 'string2'
!set permission1.maxRoles := 1
!set permission1.name := 'string2'
!set role1.maxSeniors := 1
!set role2.exclusiveJuniorsAllowed := true
!set resource2.name := 'string2'
!set permission2.maxRoles := 1
!set resource1.resourceBasedDynamicSeparationOfDuty := true
!set permission1.maxSessions := 1
!set role1.exclusiveJuniorsAllowed := false
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (role2,  role1) into PrerequisiteRoles

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': FAILED.
  -> false : Boolean
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.016s, 1 failure.
```

Permission::MaximumNumberOfSessions

```
Permission_MaximumNumberOfSessions Independence
-----------------------------------------------------------------------

reset
!create user2:User
!create session1:Session
!create session2:Session
!create role2:Role
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create snapshot2:Snapshot
!create permission2:Permission between(action2, resource1)
!set user2.maxRolesRespectingHierarchy := false
!set role2.maxSeniors := 2
!set user2.maxSessions := 2
!set resource2.historyBasedDynamicSeparationOfDuty := true
!set role2.maxJuniors := 2
!set permission2.maxSessions := 1
!set role2.exclusiveJuniorsAllowed := false
!set permission2.maxRoles := 1
!set resource1.resourceBasedDynamicSeparationOfDuty := false
!insert (permission2,  role2) into PermissionAssignment
!insert (user2,  role2) into UserAssignment
!insert (session1,  user2) into ActiveUser
```

```
!insert (session2,  user2) into ActiveUser
!insert (session1,  role2) into ActiveRoles
!insert (session2,  role2) into ActiveRoles
!insert (role2,  role2) into PrerequisiteRoles
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: FAILED.
  -> false : Boolean
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.015s, 1 failure.
```

Permission::NoPermissionAssignedtoExclusiveRoles

```
Permission_NoPermissionAssignedtoExclusiveRoles Independence
-----------------------------------------------------------------------
```

```
reset
!create role1:Role
!create role2:Role
!create action2:Action
!create resource2:Resource
!create permission2:Permission between(action2, resource2)
!create mutuallyExclusive1:MutuallyExclusive between(role2, role1)
!create mutuallyExclusive2:MutuallyExclusive between(role1, role2)
!set mutuallyExclusive2.wrtUserAssignment := true
!set role2.maxSeniors := 1
!set mutuallyExclusive1.wrtPermissionAssignment := true
!set role2.maxJuniors := 1
!set mutuallyExclusive2.wrtActiveRoles := false
!set permission2.maxSessions := 1
!set role2.exclusiveJuniorsAllowed := false
!set permission2.name := 'string2'
!set role1.name := 'string2'
!set role1.exclusiveJuniorsAllowed := true
!set mutuallyExclusive1.wrtUserAssignment := false
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': FAILED.
  -> false : Boolean
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.015s, 1 failure.
```

Permission::RequiredPermissionsPresent

```
Permission_RequiredPermissionsPresent Independence
-----------------------------------------------------------------------

reset
!create role2:Role
!create action1:Action
!create action2:Action
!create resource2:Resource
!create permission1:Permission between(action1, resource2)
!create permission2:Permission between(action2, resource2)
!set action2.name := ’string2’
!set role2.maxSeniors := 2
!set resource2.historyBasedDynamicSeparationOfDuty := true
!set resource2.resourceBasedDynamicSeparationOfDuty := true
!set permission1.maxRoles := 1
!set role2.exclusiveJuniorsAllowed := false
!set resource2.name := ’string2’
!insert (permission2,  role2) into PermissionAssignment
!insert (role2,  role2) into PrerequisiteRoles
!insert (permission1,  permission1) into PrerequisitePermissions
!insert (permission1,  permission2) into PrerequisitePermissions

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: FAILED.
```

```
  -> false : Boolean
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.032s, 1 failure.
```

Role::MaximumNumberOfJuniors

```
Role_MaximumNumberOfJuniors Independence
-----------------------------------------------------------------------

reset
!create user2:User
!create user3:User
!create role1:Role
!create role2:Role
!create role3:Role
!create action1:Action
!create action2:Action
!create action3:Action
!create resource3:Resource
!create snapshot2:Snapshot
!create snapshot3:Snapshot
!create permission3:Permission between(action1, resource3)
!create mutuallyExclusive3:MutuallyExclusive between(role2, role3)
!set user2.maxRolesRespectingHierarchy := false
!set action2.name := 'string3'
```

```
!set permission3.name := 'string3'
!set role3.exclusiveJuniorsAllowed := true
!set user3.maxRolesRespectingHierarchy := true
!set resource3.historyBasedDynamicSeparationOfDuty := false
!set role3.maxJuniors := 1
!set role3.name := 'string3'
!set mutuallyExclusive3.wrtSeniors := false
!set resource3.resourceBasedDynamicSeparationOfDuty := false
!set action3.name := 'string3'
!set role2.exclusiveJuniorsAllowed := true
!set role1.maxJuniors := 1
!set user2.maxRoles := 1
!set mutuallyExclusive3.wrtJuniors := true
!set role1.exclusiveJuniorsAllowed := true
!set resource3.name := 'string3'
!set role2.name := 'string2'
!insert (permission3,  role1) into PermissionAssignment
!insert (permission3,  role2) into PermissionAssignment
!insert (permission3,  role3) into PermissionAssignment
!insert (user2,  role3) into UserAssignment
!insert (user3,  role1) into UserAssignment
!insert (role3,  role1) into RoleHierarchy
!insert (role3,  role2) into RoleHierarchy
!insert (role2,  role2) into PrerequisiteRoles
!insert (role3,  role2) into PrerequisiteRoles
!insert (permission3,  permission3) into PrerequisitePermissions
!insert (snapshot3,  snapshot2) into PredSuccSnapshot
!insert (snapshot2,  user3) into SnapshotUser
!insert (snapshot3,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': FAILED.
  -> false : Boolean
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.032s, 1 failure.
```

Role::MaximumNumberOfMembers

```
Role_MaximumNumberOfMembers Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create user2:User
!create session2:Session
!create role2:Role
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create access2:Access
!create snapshot2:Snapshot
!create permission1:Permission between(action2, resource1)
!create permission2:Permission between(action2, resource2)
!set user2.maxRolesRespectingHierarchy := true
!set action2.name := ’string2’
!set session2.id := ’string2’
!set role2.maxMembers := 1
!set access2.id := ’string2’
!set role2.maxSeniors := 2
!set resource2.resourceBasedDynamicSeparationOfDuty := true
!set resource1.name := ’string2’
!set role2.maxJuniors := 1
```

```
!set permission1.maxRoles := 1
!set permission1.name := 'string2'
!set permission2.maxSessions := 1
!set role2.exclusiveJuniorsAllowed := true
!set permission2.maxRoles := 1
!set resource1.historyBasedDynamicSeparationOfDuty := true
!set permission1.maxSessions := 1
!set user1.maxRolesRespectingHierarchy := false
!insert (permission2,  role2) into PermissionAssignment
!insert (user1,  role2) into UserAssignment
!insert (user2,  role2) into UserAssignment
!insert (session2,  user2) into ActiveUser
!insert (session2,  role2) into ActiveRoles
!insert (session2,  access2) into ActiveAccess
!insert (access2,  action2) into AccessAction
!insert (access2,  resource2) into AccessResource
!insert (snapshot2,  user1) into SnapshotUser
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': FAILED.
  -> false : Boolean
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) `Session::SessionIdIdentifies': OK.
checking invariant (22) `Session::SuccSessionRelatedToSuccUser': OK.
checking invariant (23) `Snapshot::ChainOfSnapshots': OK.
checking invariant (24) `User::HistoryBasedDynamicSeparationOfDuty': OK.
checking invariant (25) `User::MaximumNumberOfRoles': OK.
checking invariant (26) `User::MaximumNumberOfSessions': OK.
checking invariant (27) `User::NoUserAssignedtoExclusiveRoles': OK.
checking invariant (28) `User::ResourceBasedDynamicSeparationOfDuty': OK.
checking invariant (29) `User::SuccUserInSuccSnapshot': OK.
checking invariant (30) `User::UserNameIdentifies': OK.
checked 30 invariants in 0.032s, 1 failure.
```

Role::MaximumNumberOfSeniors

```
Role_MaximumNumberOfSeniors Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create user2:User
!create user3:User
!create role1:Role
!create role2:Role
!create role3:Role
!create action3:Action
!create resource2:Resource
!create resource3:Resource
!create snapshot1:Snapshot
!create snapshot2:Snapshot
!create snapshot3:Snapshot
!create permission3:Permission between(action3, resource2)
!set user2.maxRolesRespectingHierarchy := true
!set permission3.name := 'string3'
!set role3.exclusiveJuniorsAllowed := true
!set user3.maxRolesRespectingHierarchy := false
!set user1.maxRoles := 1
!set resource3.historyBasedDynamicSeparationOfDuty := true
!set role2.maxSeniors := 1
!set role3.name := 'string3'
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set role2.maxJuniors := 1
!set user1.maxSessions := 1
!set resource3.resourceBasedDynamicSeparationOfDuty := true
!set role2.exclusiveJuniorsAllowed := false
!set resource2.name := 'string3'
!set role1.exclusiveJuniorsAllowed := true
!set user1.maxRolesRespectingHierarchy := false
!set user3.maxRoles := 1
```

```
!set resource3.name := 'string2'
!set role3.maxSeniors := 1
!set role2.name := 'string2'
!insert (permission3, role1) into PermissionAssignment
!insert (permission3, role2) into PermissionAssignment
!insert (permission3, role3) into PermissionAssignment
!insert (user1, role1) into UserAssignment
!insert (user2, role1) into UserAssignment
!insert (user2, role2) into UserAssignment
!insert (user2, role3) into UserAssignment
!insert (user3, role1) into UserAssignment
!insert (role1, role2) into RoleHierarchy
!insert (role1, role3) into RoleHierarchy
!insert (role2, role3) into RoleHierarchy
!insert (role1, role2) into PrerequisiteRoles
!insert (role2, role2) into PrerequisiteRoles
!insert (role2, role3) into PrerequisiteRoles
!insert (role3, role2) into PrerequisiteRoles
!insert (permission3, permission3) into PrerequisitePermissions
!insert (snapshot2, snapshot3) into PredSuccSnapshot
!insert (snapshot3, snapshot1) into PredSuccSnapshot
!insert (user3, user2) into PredSuccUser
!insert (snapshot1, user1) into SnapshotUser
!insert (snapshot2, user3) into SnapshotUser
!insert (snapshot3, user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) 'Access::AccessIdIdentifies': OK.
checking invariant (2) 'Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) 'MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) 'MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) 'Permission::MaximumNumberOfRoles': OK.
checking invariant (6) 'Permission::MaximumNumberOfSessions': OK.
checking invariant (7) 'Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) 'Permission::RequiredPermissionsPresent': OK.
checking invariant (9) 'Role::MaximumNumberOfJuniors': OK.
checking invariant (10) 'Role::MaximumNumberOfMembers': OK.
checking invariant (11) 'Role::MaximumNumberOfSeniors': FAILED.
  -> false : Boolean
checking invariant (12) 'Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) 'Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) 'Role::RequiredRolesNotExclusive': OK.
checking invariant (15) 'Role::RequiredRolesPresent': OK.
checking invariant (16) 'Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

Role::NoSharedJuniorsOfExclusiveRoles

```
Role_NoSharedJuniorsOfExclusiveRoles Independence
-----------------------------------------------------------------------

reset
!create user3:User
!create session3:Session
!create role1:Role
!create role2:Role
!create role3:Role
!create action3:Action
!create resource2:Resource
!create resource3:Resource
!create access1:Access
!create access2:Access
!create access3:Access
!create snapshot3:Snapshot
!create permission2:Permission between(action3, resource2)
!create permission3:Permission between(action3, resource3)
!create mutuallyExclusive1:MutuallyExclusive between(role1, role3)
!create mutuallyExclusive2:MutuallyExclusive between(role2, role3)
!create mutuallyExclusive3:MutuallyExclusive between(role1, role2)
!set mutuallyExclusive3.identicalSeniorAllowed := true
!set mutuallyExclusive1.wrtJuniors := true
!set mutuallyExclusive2.identicalSeniorAllowed := true
!set mutuallyExclusive3.wrtUserAssignment := true
!set access1.id := ’string3’
!set role3.exclusiveJuniorsAllowed := true
!set resource3.historyBasedDynamicSeparationOfDuty := false
```

```
!set role3.maxJuniors := 1
!set mutuallyExclusive1.identicalSeniorAllowed := false
!set access2.id := 'string3'
!set resource2.historyBasedDynamicSeparationOfDuty := true
!set mutuallyExclusive2.wrtJuniors := false
!set mutuallyExclusive3.wrtSeniors := false
!set role2.exclusiveJuniorsAllowed := true
!set permission2.maxRoles := 1
!set role1.name := 'string3'
!set mutuallyExclusive2.wrtSeniors := true
!set mutuallyExclusive3.wrtActiveRoles := true
!set access3.id := 'string3'
!set role1.exclusiveJuniorsAllowed := true
!set user3.maxRoles := 2
!set mutuallyExclusive1.wrtUserAssignment := false
!set role3.maxSeniors := 1
!insert (permission2,  role3) into PermissionAssignment
!insert (permission3,  role1) into PermissionAssignment
!insert (permission3,  role2) into PermissionAssignment
!insert (user3,  role2) into UserAssignment
!insert (session3,  user3) into ActiveUser
!insert (session3,  role2) into ActiveRoles
!insert (role1,  role2) into RoleHierarchy
!insert (role3,  role2) into RoleHierarchy
!insert (session3,  access1) into ActiveAccess
!insert (session3,  access2) into ActiveAccess
!insert (session3,  access3) into ActiveAccess
!insert (access1,  action3) into AccessAction
!insert (access2,  action3) into AccessAction
!insert (access3,  action3) into AccessAction
!insert (access1,  resource3) into AccessResource
!insert (access2,  resource3) into AccessResource
!insert (access3,  resource3) into AccessResource
!insert (snapshot3,  user3) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: FAILED.
  -> false : Boolean
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

Role::NoSharedSeniorsOfExclusiveRoles

```
Role_NoSharedSeniorsOfExclusiveRoles Independence
-----------------------------------------------------------------------

reset
!create user3:User
!create session1:Session
!create session2:Session
!create session3:Session
!create role1:Role
!create role2:Role
!create role3:Role
!create action2:Action
!create action3:Action
!create resource1:Resource
!create resource2:Resource
!create resource3:Resource
!create snapshot1:Snapshot
!create snapshot2:Snapshot
!create snapshot3:Snapshot
```

```
!create permission2:Permission between(action2, resource3)
!create permission3:Permission between(action3, resource1)
!create mutuallyExclusive3:MutuallyExclusive between(role3, role1)
!set mutuallyExclusive3.identicalSeniorAllowed := false
!set session1.id := 'string3'
!set mutuallyExclusive3.wrtUserAssignment := false
!set action2.name := 'string3'
!set permission3.name := 'string2'
!set role3.exclusiveJuniorsAllowed := false
!set user3.maxRolesRespectingHierarchy := false
!set resource3.historyBasedDynamicSeparationOfDuty := true
!set session2.id := 'string2'
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource2.resourceBasedDynamicSeparationOfDuty := true
!set resource1.name := 'string2'
!set mutuallyExclusive3.wrtSeniors := true
!set resource3.resourceBasedDynamicSeparationOfDuty := false
!set action3.name := 'string2'
!set role2.exclusiveJuniorsAllowed := true
!set mutuallyExclusive3.wrtJuniors := false
!set resource2.name := 'string3'
!set permission2.name := 'string3'
!set role1.name := 'string2'
!set mutuallyExclusive3.wrtActiveRoles := false
!set resource1.resourceBasedDynamicSeparationOfDuty := false
!set role1.exclusiveJuniorsAllowed := true
!set mutuallyExclusive3.wrtPermissionAssignment := false
!set resource3.name := 'string2'
!set role2.name := 'string2'
!insert (permission2,  role2) into PermissionAssignment
!insert (permission2,  role3) into PermissionAssignment
!insert (permission3,  role1) into PermissionAssignment
!insert (user3,  role1) into UserAssignment
!insert (user3,  role2) into UserAssignment
!insert (user3,  role3) into UserAssignment
!insert (session1,  user3) into ActiveUser
!insert (session2,  user3) into ActiveUser
!insert (session3,  user3) into ActiveUser
!insert (session1,  role3) into ActiveRoles
!insert (session2,  role3) into ActiveRoles
!insert (session3,  role2) into ActiveRoles
!insert (role1,  role3) into RoleHierarchy
!insert (role2,  role1) into RoleHierarchy
!insert (role2,  role3) into RoleHierarchy
!insert (role1,  role3) into PrerequisiteRoles
```

```
!insert (role3,  role1) into PrerequisiteRoles
!insert (role3,  role2) into PrerequisiteRoles
!insert (role3,  role3) into PrerequisiteRoles
!insert (permission2,  permission2) into PrerequisitePermissions
!insert (snapshot2,  snapshot3) into PredSuccSnapshot
!insert (snapshot3,  snapshot1) into PredSuccSnapshot
!insert (snapshot1,  user3) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: FAILED.
  -> false : Boolean
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

Role::RequiredRolesNotExclusive

```
Role_RequiredRolesNotExclusive Independence
```

```
----------------------------------------------------------------------

reset
!create user2:User
!create session2:Session
!create role1:Role
!create role2:Role
!create action1:Action
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create access2:Access
!create snapshot2:Snapshot
!create permission1:Permission between(action2, resource2)
!create permission2:Permission between(action1, resource1)
!create mutuallyExclusive2:MutuallyExclusive between(role2, role1)
!set action2.name := 'string2'
!set mutuallyExclusive2.wrtUserAssignment := true
!set access2.id := 'string2'
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource1.name := 'string2'
!set mutuallyExclusive2.wrtJuniors := false
!set permission1.name := 'string2'
!set mutuallyExclusive2.wrtActiveRoles := false
!set role2.exclusiveJuniorsAllowed := true
!set mutuallyExclusive2.wrtPermissionAssignment := false
!set role1.name := 'string2'
!set mutuallyExclusive2.wrtSeniors := false
!set role1.exclusiveJuniorsAllowed := true
!insert (permission1,  role2) into PermissionAssignment
!insert (permission2,  role1) into PermissionAssignment
!insert (user2,  role1) into UserAssignment
!insert (session2,  user2) into ActiveUser
!insert (session2,  role1) into ActiveRoles
!insert (role1,  role2) into RoleHierarchy
!insert (role1,  role2) into PrerequisiteRoles
!insert (session2,  access2) into ActiveAccess
!insert (access2,  action2) into AccessAction
!insert (access2,  resource2) into AccessResource
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: FAILED.
  -> false : Boolean
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.015s, 1 failure.
```

Role::RequiredRolesPresent

```
Role_RequiredRolesPresent Independence
-----------------------------------------------------------------------

reset
!create user2:User
!create role1:Role
!create role2:Role
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create snapshot2:Snapshot
!create permission2:Permission between(action2, resource1)
!create mutuallyExclusive1:MutuallyExclusive between(role1, role2)
```

```
!create mutuallyExclusive2:MutuallyExclusive between(role2, role1)
!set user2.maxRolesRespectingHierarchy := false
!set mutuallyExclusive1.wrtJuniors := true
!set mutuallyExclusive2.identicalSeniorAllowed := false
!set action2.name := 'string2'
!set user2.maxSessions := 1
!set resource2.historyBasedDynamicSeparationOfDuty := true
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set role2.maxJuniors := 1
!set mutuallyExclusive2.wrtJuniors := false
!set permission2.maxSessions := 1
!set role2.exclusiveJuniorsAllowed := true
!set permission2.name := 'string1'
!set role1.name := 'string1'
!set mutuallyExclusive2.wrtSeniors := true
!set role1.exclusiveJuniorsAllowed := true
!set role2.name := 'string1'
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (user2,  role2) into UserAssignment
!insert (role1,  role2) into PrerequisiteRoles
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': FAILED.
  -> false : Boolean
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

Role::RoleHierarchyPartialOrder

```
Role_RoleHierarchyPartialOrder Independence
-----------------------------------------------------------------------

reset
!create role1:Role
!create action1:Action
!create resource1:Resource
!create permission1:Permission between(action1, resource1)
!insert (permission1,  role1) into PermissionAssignment
!insert (role1,  role1) into RoleHierarchy

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: FAILED.
  -> false : Boolean
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.016s, 1 failure.
```

Role::SeniorsWithExclusiveJuniors

```
Role_SeniorsWithExclusiveJuniors Independence
-----------------------------------------------------------------------

reset
!create role1:Role
!create role2:Role
!create action2:Action
!create resource2:Resource
!create permission2:Permission between(action2, resource2)
!create mutuallyExclusive2:MutuallyExclusive between(role2, role1)
!set mutuallyExclusive2.wrtUserAssignment := true
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set mutuallyExclusive2.wrtJuniors := false
!set role2.exclusiveJuniorsAllowed := false
!set resource2.name := ’string2’
!set permission2.name := ’string2’
!set role1.exclusiveJuniorsAllowed := true
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (role2,  role1) into RoleHierarchy
!insert (permission2,  permission2) into PrerequisitePermissions

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: FAILED.
  -> false : Boolean
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.015s, 1 failure.
```

Session::ActionsPermitted

```
Session_ActionsPermitted.txt Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create session1:Session
!create role1:Role
!create action1:Action
!create resource1:Resource
!create access1:Access
!create snapshot1:Snapshot
!create permission1:Permission between(action1, resource1)
!insert (permission1,  role1) into PermissionAssignment
!insert (user1,  role1) into UserAssignment
!insert (session1,  user1) into ActiveUser
```

```
!insert (session1,  access1) into ActiveAccess
!insert (access1,  action1) into AccessAction
!insert (access1,  resource1) into AccessResource
!insert (snapshot1,  user1) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': FAILED.
  -> false : Boolean
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) `Session::SessionIdIdentifies': OK.
checking invariant (22) `Session::SuccSessionRelatedToSuccUser': OK.
checking invariant (23) `Snapshot::ChainOfSnapshots': OK.
checking invariant (24) `User::HistoryBasedDynamicSeparationOfDuty': OK.
checking invariant (25) `User::MaximumNumberOfRoles': OK.
checking invariant (26) `User::MaximumNumberOfSessions': OK.
checking invariant (27) `User::NoUserAssignedtoExclusiveRoles': OK.
checking invariant (28) `User::ResourceBasedDynamicSeparationOfDuty': OK.
checking invariant (29) `User::SuccUserInSuccSnapshot': OK.
checking invariant (30) `User::UserNameIdentifies': OK.
checked 30 invariants in 0.016s, 1 failure.
```

Session::ActiveRolesSubsetUserRoles

```
Session_ActiveRolesSubsetUserRoles Independence
-----------------------------------------------------------------------

reset
```

```
!create user1:User
!create user2:User
!create session2:Session
!create role1:Role
!create role2:Role
!create action2:Action
!create resource2:Resource
!create snapshot2:Snapshot
!create permission2:Permission between(action2, resource2)
!set user1.maxRoles := 2
!set role2.maxMembers := 2
!set role2.maxSeniors := 2
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set role2.maxJuniors := 1
!set user1.maxSessions := 1
!set permission2.maxSessions := 2
!set role2.exclusiveJuniorsAllowed := false
!set role1.maxJuniors := 1
!set permission2.maxRoles := 2
!set user1.maxRolesRespectingHierarchy := false
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (user1,  role2) into UserAssignment
!insert (user2,  role2) into UserAssignment
!insert (session2,  user2) into ActiveUser
!insert (session2,  role1) into ActiveRoles
!insert (session2,  role2) into ActiveRoles
!insert (snapshot2,  user1) into SnapshotUser
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: FAILED.
  -> false : Boolean
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

Session::NoExclusiveRolesActive

```
Session_NoExclusiveRolesActive Independence
-----------------------------------------------------------------------

reset
!create user2:User
!create session2:Session
!create role1:Role
!create role2:Role
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create snapshot2:Snapshot
!create permission1:Permission between(action2, resource2)
!create permission2:Permission between(action2, resource1)
!create mutuallyExclusive2:MutuallyExclusive between(role2, role1)
!set user2.maxRolesRespectingHierarchy := false
!set mutuallyExclusive2.identicalSeniorAllowed := false
!set role2.maxSeniors := 1
!set mutuallyExclusive2.wrtJuniors := false
!set mutuallyExclusive2.wrtActiveRoles := true
!set role2.exclusiveJuniorsAllowed := true
!set mutuallyExclusive2.wrtPermissionAssignment := false
!set permission2.maxRoles := 1
```

```
!set permission1.maxSessions := 1
!set role1.exclusiveJuniorsAllowed := true
!insert (permission1,  role2) into PermissionAssignment
!insert (permission2,  role1) into PermissionAssignment
!insert (user2,  role1) into UserAssignment
!insert (session2,  user2) into ActiveUser
!insert (session2,  role1) into ActiveRoles
!insert (session2,  role2) into ActiveRoles
!insert (role1,  role2) into RoleHierarchy
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: FAILED.
  -> false : Boolean
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.047s, 1 failure.
```

Session::SessionIdIdentifies

```
Session_SessionIdIdentifies Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create user2:User
!create session1:Session
!create session2:Session
!create role2:Role
!create action1:Action
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create snapshot1:Snapshot
!create snapshot2:Snapshot
!create permission2:Permission between(action1, resource1)
!set user2.maxRolesRespectingHierarchy := false
!set action2.name := 'string2'
!set session2.id := 'string2'
!set user1.maxSessions := 1
!set role2.exclusiveJuniorsAllowed := true
!set resource2.name := 'string2'
!set permission2.name := 'string2'
!set resource1.resourceBasedDynamicSeparationOfDuty := true
!insert (permission2,  role2) into PermissionAssignment
!insert (user1,  role2) into UserAssignment
!insert (user2,  role2) into UserAssignment
!insert (session1,  user2) into ActiveUser
!insert (session2,  user1) into ActiveUser
!insert (permission2,  permission2) into PrerequisitePermissions
!insert (snapshot2,  snapshot1) into PredSuccSnapshot
!insert (user1,  user2) into PredSuccUser
!insert (session2,  session1) into PredSuccSession
!insert (snapshot1,  user2) into SnapshotUser
!insert (snapshot2,  user1) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: FAILED.
  -> false : Boolean
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

Session::SuccSessionRelatedToSuccUser

```
Session_SuccSessionRelatedToSuccUser Independence
-----------------------------------------------------------------------

reset
!create user2:User
!create session1:Session
!create session2:Session
!create role2:Role
!create action1:Action
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create snapshot2:Snapshot
!create permission1:Permission between(action2, resource2)
!create permission2:Permission between(action2, resource1)
!set action1.name := ’string2’
!set user2.maxRolesRespectingHierarchy := true
```

```
!set action2.name := 'string2'
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource1.name := 'string2'
!set permission1.name := 'string2'
!set role2.exclusiveJuniorsAllowed := false
!set resource2.name := 'string2'
!set permission2.name := 'string2'
!set resource1.resourceBasedDynamicSeparationOfDuty := false
!set role2.name := 'string2'
!insert (permission2,  role2) into PermissionAssignment
!insert (user2,  role2) into UserAssignment
!insert (session1,  user2) into ActiveUser
!insert (session2,  user2) into ActiveUser
!insert (session1,  role2) into ActiveRoles
!insert (session2,  role2) into ActiveRoles
!insert (role2,  role2) into PrerequisiteRoles
!insert (permission2,  permission1) into PrerequisitePermissions
!insert (permission2,  permission2) into PrerequisitePermissions
!insert (session2,  session1) into PredSuccSession
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) `Session::SessionIdIdentifies': OK.
checking invariant (22) `Session::SuccSessionRelatedToSuccUser': FAILED.
```

```
  -> false : Boolean
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

Snapshot::ChainOfSnapshots

```
Snapshot_ChainOfSnapshots Independence
-----------------------------------------------------------------------

reset
!create action1:Action
!create resource1:Resource
!create snapshot1:Snapshot
!create permission1:Permission between(action1, resource1)
!insert (snapshot1,  snapshot1) into PredSuccSnapshot

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: FAILED.
  -> false : Boolean
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

User::HistoryBasedDynamicSeparationOfDuty

```
User_HistoryBasedDynamicSeparationOfDuty Independence
-----------------------------------------------------------------------

reset
!create user2:User
!create session1:Session
!create session2:Session
!create role1:Role
!create role2:Role
!create action1:Action
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create access1:Access
!create access2:Access
!create snapshot2:Snapshot
!create permission1:Permission between(action2, resource2)
!create permission2:Permission between(action1, resource2)
!create mutuallyExclusive2:MutuallyExclusive between(role1, role2)
!set user2.maxRolesRespectingHierarchy := true
!set session1.id := ’string2’
!set mutuallyExclusive2.identicalSeniorAllowed := true
!set access1.id := ’string2’
!set mutuallyExclusive2.wrtUserAssignment := true
!set resource2.historyBasedDynamicSeparationOfDuty := true
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set role2.maxJuniors := 1
!set role1.maxMembers := 1
!set permission2.maxSessions := 1
!set role2.exclusiveJuniorsAllowed := true
!set resource1.historyBasedDynamicSeparationOfDuty := false
!set resource1.resourceBasedDynamicSeparationOfDuty := true
```

```
!set role1.exclusiveJuniorsAllowed := false
!insert (permission1,  role2) into PermissionAssignment
!insert (permission2,  role1) into PermissionAssignment
!insert (user2,  role1) into UserAssignment
!insert (session1,  user2) into ActiveUser
!insert (session2,  user2) into ActiveUser
!insert (session1,  role1) into ActiveRoles
!insert (session1,  role2) into ActiveRoles
!insert (session2,  role2) into ActiveRoles
!insert (role1,  role2) into RoleHierarchy
!insert (role2,  role2) into PrerequisiteRoles
!insert (permission1,  permission1) into PrerequisitePermissions
!insert (session1,  access2) into ActiveAccess
!insert (session2,  access1) into ActiveAccess
!insert (access1,  action2) into AccessAction
!insert (access2,  action1) into AccessAction
!insert (access1,  resource2) into AccessResource
!insert (access2,  resource2) into AccessResource
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) `Session::SessionIdIdentifies': OK.
checking invariant (22) `Session::SuccSessionRelatedToSuccUser': OK.
checking invariant (23) `Snapshot::ChainOfSnapshots': OK.
checking invariant (24) `User::HistoryBasedDynamicSeparationOfDuty': FAILED.
  -> false : Boolean
checking invariant (25) `User::MaximumNumberOfRoles': OK.
checking invariant (26) `User::MaximumNumberOfSessions': OK.
checking invariant (27) `User::NoUserAssignedtoExclusiveRoles': OK.
checking invariant (28) `User::ResourceBasedDynamicSeparationOfDuty': OK.
checking invariant (29) `User::SuccUserInSuccSnapshot': OK.
checking invariant (30) `User::UserNameIdentifies': OK.
checked 30 invariants in 0.016s, 1 failure.
```

User::MaximumNumberOfRoles

```
User_MaximumNumberOfRoles Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create user2:User
!create role1:Role
!create role2:Role
!create action1:Action
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create snapshot1:Snapshot
!create snapshot2:Snapshot
!create permission1:Permission between(action2, resource1)
!create permission2:Permission between(action1, resource2)
!set action1.name := 'string2'
!set user2.maxRolesRespectingHierarchy := true
!set action2.name := 'string1'
!set role2.maxSeniors := 2
!set resource2.historyBasedDynamicSeparationOfDuty := true
!set role2.maxJuniors := 2
!set permission1.maxRoles := 1
!set permission1.name := 'string1'
!set user1.maxSessions := 1
!set role2.exclusiveJuniorsAllowed := false
!set user2.maxRoles := 1
!set user2.name := 'string1'
!set permission2.maxRoles := 1
!set resource1.historyBasedDynamicSeparationOfDuty := false
!set permission1.maxSessions := 2
!set role1.exclusiveJuniorsAllowed := true
!set user1.maxRolesRespectingHierarchy := false
!insert (permission1,  role2) into PermissionAssignment
```

```
!insert (permission2,  role1) into PermissionAssignment
!insert (user1,  role2) into UserAssignment
!insert (user2,  role2) into UserAssignment
!insert (role2,  role1) into RoleHierarchy
!insert (permission1,  permission1) into PrerequisitePermissions
!insert (snapshot2,  snapshot1) into PredSuccSnapshot
!insert (snapshot1,  user2) into SnapshotUser
!insert (snapshot2,  user1) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: FAILED.
  -> false : Boolean
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.032s, 1 failure.
```

User::MaximumNumberOfSessions

```
User_MaximumNumberOfSessions Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create user2:User
!create session1:Session
!create session2:Session
!create role2:Role
!create action2:Action
!create resource2:Resource
!create snapshot2:Snapshot
!create permission2:Permission between(action2, resource2)
!set user2.maxSessions := 1
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set user1.maxSessions := 1
!set permission2.maxSessions := 2
!set role2.exclusiveJuniorsAllowed := true
!set user2.maxRoles := 1
!set permission2.maxRoles := 1
!insert (permission2,  role2) into PermissionAssignment
!insert (user1,  role2) into UserAssignment
!insert (user2,  role2) into UserAssignment
!insert (session1,  user2) into ActiveUser
!insert (session2,  user2) into ActiveUser
!insert (session2,  role2) into ActiveRoles
!insert (permission2,  permission2) into PrerequisitePermissions
!insert (snapshot2,  user1) into SnapshotUser
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: FAILED.
  -> false : Boolean
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.032s, 1 failure.
```

User::NoUserAssignedtoExclusiveRoles

```
User_NoUserAssignedtoExclusiveRoles Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create user2:User
!create session2:Session
!create role1:Role
!create role2:Role
!create action2:Action
!create resource1:Resource
!create resource2:Resource
!create snapshot2:Snapshot
!create permission2:Permission between(action2, resource1)
!create mutuallyExclusive2:MutuallyExclusive between(role1, role2)
!set mutuallyExclusive2.identicalSeniorAllowed := false
!set user1.maxRoles := 1
!set user1.name := ’string2’
!set mutuallyExclusive2.wrtUserAssignment := true
!set resource2.historyBasedDynamicSeparationOfDuty := true
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set mutuallyExclusive2.wrtJuniors := false
!set role1.maxSeniors := 1
!set mutuallyExclusive2.wrtActiveRoles := false
```

```
!set role2.exclusiveJuniorsAllowed := true
!set mutuallyExclusive2.wrtPermissionAssignment := false
!set resource2.name := 'string1'
!set permission2.name := 'string2'
!set mutuallyExclusive2.wrtSeniors := false
!set role1.exclusiveJuniorsAllowed := false
!set role2.name := 'string1'
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (user1,  role2) into UserAssignment
!insert (user2,  role1) into UserAssignment
!insert (user2,  role2) into UserAssignment
!insert (session2,  user1) into ActiveUser
!insert (session2,  role2) into ActiveRoles
!insert (role1,  role1) into PrerequisiteRoles
!insert (snapshot2,  user1) into SnapshotUser
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) `Session::SessionIdIdentifies': OK.
checking invariant (22) `Session::SuccSessionRelatedToSuccUser': OK.
checking invariant (23) `Snapshot::ChainOfSnapshots': OK.
checking invariant (24) `User::HistoryBasedDynamicSeparationOfDuty': OK.
checking invariant (25) `User::MaximumNumberOfRoles': OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: FAILED.
  -> false : Boolean
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.031s, 1 failure.
```

User::ResourceBasedDynamicSeparationOfDuty

```
User_ResourceBasedDynamicSeparationOfDuty Independence
-----------------------------------------------------------------------

reset
!create user2:User
!create session2:Session
!create role2:Role
!create action1:Action
!create action2:Action
!create resource2:Resource
!create access1:Access
!create access2:Access
!create snapshot2:Snapshot
!create permission1:Permission between(action1, resource2)
!create permission2:Permission between(action2, resource2)
!set user2.maxRolesRespectingHierarchy := false
!set role2.maxSeniors := 2
!set user2.maxSessions := 2
!set resource2.historyBasedDynamicSeparationOfDuty := false
!set resource2.resourceBasedDynamicSeparationOfDuty := true
!set role2.maxJuniors := 1
!set role2.exclusiveJuniorsAllowed := true
!set user2.maxRoles := 1
!set permission2.maxRoles := 2
!insert (permission1,  role2) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (user2,  role2) into UserAssignment
!insert (session2,  user2) into ActiveUser
!insert (session2,  role2) into ActiveRoles
!insert (role2,  role2) into PrerequisiteRoles
!insert (permission2,  permission1) into PrerequisitePermissions
!insert (permission2,  permission2) into PrerequisitePermissions
!insert (session2,  access1) into ActiveAccess
!insert (session2,  access2) into ActiveAccess
!insert (access1,  action2) into AccessAction
!insert (access2,  action1) into AccessAction
```

```
!insert (access1,  resource2) into AccessResource
!insert (access2,  resource2) into AccessResource
!insert (snapshot2,  user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) `Access::AccessIdIdentifies': OK.
checking invariant (2) `Access::SuccAccessRelatedToSuccSession': OK.
checking invariant (3) `MutuallyExclusive::DeterminationOfAtLeastOneExclusion': OK.
checking invariant (4) `MutuallyExclusive::NoSelfExclusion': OK.
checking invariant (5) `Permission::MaximumNumberOfRoles': OK.
checking invariant (6) `Permission::MaximumNumberOfSessions': OK.
checking invariant (7) `Permission::NoPermissionAssignedtoExclusiveRoles': OK.
checking invariant (8) `Permission::RequiredPermissionsPresent': OK.
checking invariant (9) `Role::MaximumNumberOfJuniors': OK.
checking invariant (10) `Role::MaximumNumberOfMembers': OK.
checking invariant (11) `Role::MaximumNumberOfSeniors': OK.
checking invariant (12) `Role::NoSharedJuniorsOfExclusiveRoles': OK.
checking invariant (13) `Role::NoSharedSeniorsOfExclusiveRoles': OK.
checking invariant (14) `Role::RequiredRolesNotExclusive': OK.
checking invariant (15) `Role::RequiredRolesPresent': OK.
checking invariant (16) `Role::RoleHierarchyPartialOrder': OK.
checking invariant (17) `Role::SeniorsWithExclusiveJuniors': OK.
checking invariant (18) `Session::ActionsPermitted': OK.
checking invariant (19) `Session::ActiveRolesSubsetUserRoles': OK.
checking invariant (20) `Session::NoExclusiveRolesActive': OK.
checking invariant (21) `Session::SessionIdIdentifies': OK.
checking invariant (22) `Session::SuccSessionRelatedToSuccUser': OK.
checking invariant (23) `Snapshot::ChainOfSnapshots': OK.
checking invariant (24) `User::HistoryBasedDynamicSeparationOfDuty': OK.
checking invariant (25) `User::MaximumNumberOfRoles': OK.
checking invariant (26) `User::MaximumNumberOfSessions': OK.
checking invariant (27) `User::NoUserAssignedtoExclusiveRoles': OK.
checking invariant (28) `User::ResourceBasedDynamicSeparationOfDuty': FAILED.
  -> false : Boolean
checking invariant (29) `User::SuccUserInSuccSnapshot': OK.
checking invariant (30) `User::UserNameIdentifies': OK.
checked 30 invariants in 0.031s, 1 failure.
```

User::SuccUserInSuccSnapshot

```
User_SuccUserInSuccSnapshot Independence
-----------------------------------------------------------------------

reset
!create user1:User
```

```
!create role1:Role
!create action1:Action
!create resource1:Resource
!create snapshot1:Snapshot
!create permission1:Permission between(action1, resource1)
!insert (permission1,  role1) into PermissionAssignment
!insert (user1,  role1) into UserAssignment
!insert (user1,  user1) into PredSuccUser
!insert (snapshot1,  user1) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: FAILED.
  -> false : Boolean
checking invariant (30) ‘User::UserNameIdentifies’: OK.
checked 30 invariants in 0.016s, 1 failure.
```

User::UserNameIdentifies

```
User_UserNameIdentifies Independence
-----------------------------------------------------------------------

reset
!create user1:User
!create user2:User
!create session2:Session
!create role1:Role
!create role2:Role
!create action2:Action
!create resource2:Resource
!create snapshot1:Snapshot
!create snapshot2:Snapshot
!create permission2:Permission between(action2, resource2)
!create mutuallyExclusive2:MutuallyExclusive between(role1, role2)
!set user2.maxRolesRespectingHierarchy := false
!set mutuallyExclusive2.identicalSeniorAllowed := false
!set action2.name := 'string2'
!set user1.name := 'string2'
!set session2.id := 'string2'
!set mutuallyExclusive2.wrtUserAssignment := false
!set resource2.historyBasedDynamicSeparationOfDuty := true
!set resource2.resourceBasedDynamicSeparationOfDuty := false
!set mutuallyExclusive2.wrtJuniors := false
!set user1.maxSessions := 1
!set mutuallyExclusive2.wrtActiveRoles := true
!set role2.exclusiveJuniorsAllowed := false
!set role1.maxJuniors := 1
!set user2.maxRoles := 1
!set mutuallyExclusive2.wrtPermissionAssignment := false
!set resource2.name := 'string2'
!set permission2.name := 'string2'
!set mutuallyExclusive2.wrtSeniors := false
!set role1.exclusiveJuniorsAllowed := true
!set user1.maxRolesRespectingHierarchy := true
!insert (permission2,  role1) into PermissionAssignment
!insert (permission2,  role2) into PermissionAssignment
!insert (user1,  role1) into UserAssignment
!insert (user2,  role1) into UserAssignment
!insert (session2,  user1) into ActiveUser
!insert (role1,  role2) into RoleHierarchy
!insert (role1,  role2) into PrerequisiteRoles
!insert (role2,  role2) into PrerequisiteRoles
!insert (snapshot2,  snapshot1) into PredSuccSnapshot
```

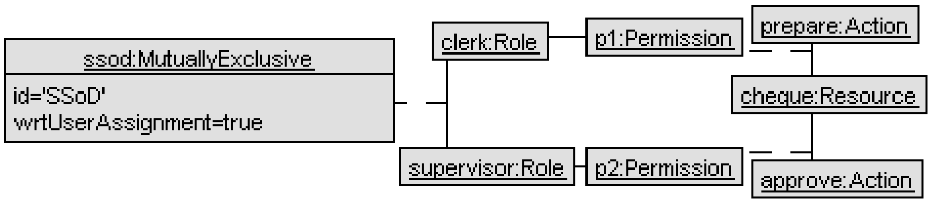
```
!insert (user2, user1) into PredSuccUser
!insert (snapshot1, user1) into SnapshotUser
!insert (snapshot2, user2) into SnapshotUser

checking structure...
checking invariants...
checking invariant (1) ‘Access::AccessIdIdentifies’: OK.
checking invariant (2) ‘Access::SuccAccessRelatedToSuccSession’: OK.
checking invariant (3) ‘MutuallyExclusive::DeterminationOfAtLeastOneExclusion’: OK.
checking invariant (4) ‘MutuallyExclusive::NoSelfExclusion’: OK.
checking invariant (5) ‘Permission::MaximumNumberOfRoles’: OK.
checking invariant (6) ‘Permission::MaximumNumberOfSessions’: OK.
checking invariant (7) ‘Permission::NoPermissionAssignedtoExclusiveRoles’: OK.
checking invariant (8) ‘Permission::RequiredPermissionsPresent’: OK.
checking invariant (9) ‘Role::MaximumNumberOfJuniors’: OK.
checking invariant (10) ‘Role::MaximumNumberOfMembers’: OK.
checking invariant (11) ‘Role::MaximumNumberOfSeniors’: OK.
checking invariant (12) ‘Role::NoSharedJuniorsOfExclusiveRoles’: OK.
checking invariant (13) ‘Role::NoSharedSeniorsOfExclusiveRoles’: OK.
checking invariant (14) ‘Role::RequiredRolesNotExclusive’: OK.
checking invariant (15) ‘Role::RequiredRolesPresent’: OK.
checking invariant (16) ‘Role::RoleHierarchyPartialOrder’: OK.
checking invariant (17) ‘Role::SeniorsWithExclusiveJuniors’: OK.
checking invariant (18) ‘Session::ActionsPermitted’: OK.
checking invariant (19) ‘Session::ActiveRolesSubsetUserRoles’: OK.
checking invariant (20) ‘Session::NoExclusiveRolesActive’: OK.
checking invariant (21) ‘Session::SessionIdIdentifies’: OK.
checking invariant (22) ‘Session::SuccSessionRelatedToSuccUser’: OK.
checking invariant (23) ‘Snapshot::ChainOfSnapshots’: OK.
checking invariant (24) ‘User::HistoryBasedDynamicSeparationOfDuty’: OK.
checking invariant (25) ‘User::MaximumNumberOfRoles’: OK.
checking invariant (26) ‘User::MaximumNumberOfSessions’: OK.
checking invariant (27) ‘User::NoUserAssignedtoExclusiveRoles’: OK.
checking invariant (28) ‘User::ResourceBasedDynamicSeparationOfDuty’: OK.
checking invariant (29) ‘User::SuccUserInSuccSnapshot’: OK.
checking invariant (30) ‘User::UserNameIdentifies’: FAILED.
  -> false : Boolean
checked 30 invariants in 0.032s, 1 failure.
```

4 Analyzing RBAC Policies

4.1 Static Separation of Duty (wrtUserAssignment)

The partial policy is determined by explicit bound definitions in Kodkod. It is based on the following object diagram:



The diagram is defined with the following USE command sequence:

```
!create cheque : Resource
!create prepare,approve : Action
!create clerk,supervisor : Role
!create p1:Permission between (prepare,cheque)
!create p2:Permission between (approve,cheque)
!insert (p1,clerk) into PermissionAssignment
!insert (p2,supervisor) into PermissionAssignment
!create ssod : MutuallyExclusive between (clerk,supervisor)
!set ssod.id := 'SSoD'
!set ssod.wrtUserAssignment := true
```

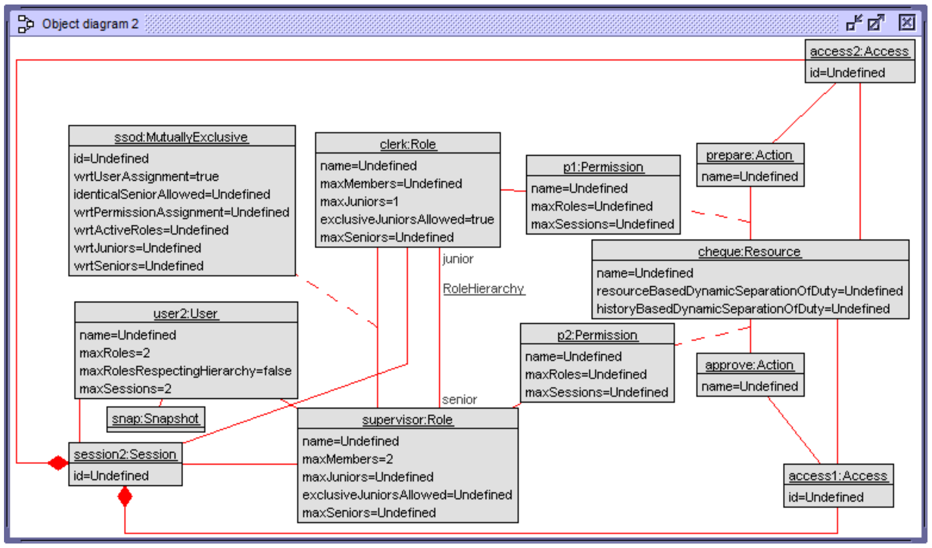
Given property to be checked:

```
Snapshot.allInstances()->size() = 1 and
User.allInstances()->exists(u|u.session.access.action->asSet()->size()=2)
```

The same constraint in relational logic:

```
one Snapshot &&
(some u: User |
  #(((ActiveUser . u) . ActiveAccess) . AccessAction) = 2)
```

OCL2Kodkod returns the following complete and valid object diagram:

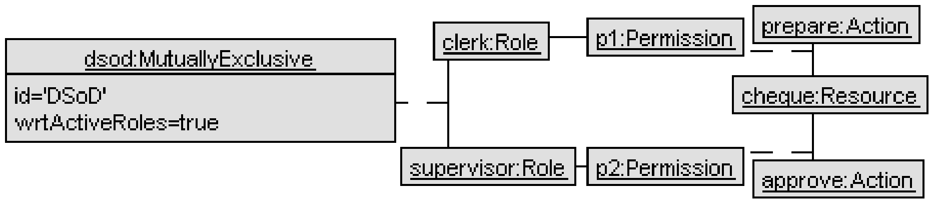


It results from the following USE command sequence which is automatically generated by OCL2Kodkod:

```
reset
!create user2:User
!create session2:Session
!create clerk:Role
!create supervisor:Role
!create prepare:Action
!create approve:Action
!create cheque:Resource
!create access1:Access
!create access2:Access
!create snap:Snapshot
!create p1:Permission between(prepare, cheque)
!create p2:Permission between(approve, cheque)
!create ssod:MutuallyExclusive between(clerk, supervisor)
!set clerk.exclusiveJuniorsAllowed := true
!set user2.maxRolesRespectingHierarchy := false
!set ssod.wrtUserAssignment := true
!set clerk.maxJuniors := 1
!set user2.maxSessions := 2
!set user2.maxRoles := 2
!set supervisor.maxMembers := 2
!insert (p1,  clerk) into PermissionAssignment
!insert (p2,  supervisor) into PermissionAssignment
!insert (user2,  supervisor) into UserAssignment
!insert (session2,  user2) into ActiveUser
!insert (session2,  clerk) into ActiveRoles
!insert (session2,  supervisor) into ActiveRoles
!insert (supervisor, clerk) into RoleHierarchy
!insert (session2,  access1) into ActiveAccess
!insert (session2,  access2) into ActiveAccess
!insert (access1,  approve) into AccessAction
!insert (access2,  prepare) into AccessAction
!insert (access1,  cheque) into AccessResource
!insert (access2,  cheque) into AccessResource
!insert (snap,  user2) into SnapshotUser
```

4.2 Dynamic Separation of Duty (wrtActiveRoles)

The partial policy is determined by explicit bound definitions in Kodkod. It is based on the following object diagram:



The diagram is defined with the following USE command sequence:

```
!create snap : Snapshot
!create cheque : Resource
!create prepare,approve : Action
!create clerk,supervisor : Role
!create p1:Permission between (prepare,cheque)
!create p2:Permission between (approve,cheque)
!insert (p1,clerk) into PermissionAssignment
!insert (p2,supervisor) into PermissionAssignment
!insert (supervisor,clerk) into RoleHierarchy
!create dsod : MutuallyExclusive between (clerk,supervisor)
!set dsod.id := 'DSoD'
!set dsod.wrtActiveRoles := true
```

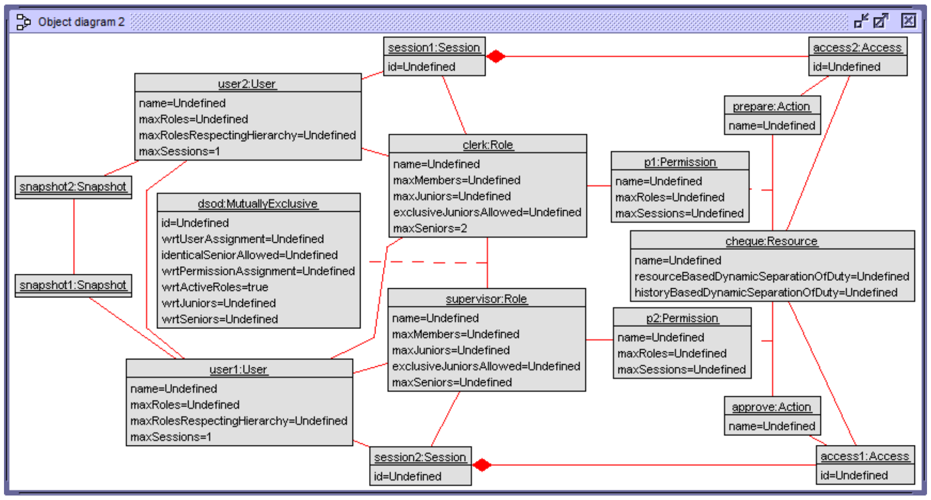
Given property to be checked:

```
User.allInstances()->exists(u|
  u.successors()->including(u)=User.allInstances()) and
User.allInstances()->exists(u|
  u.successors()->including(u).session.access.action->asSet()->size()=2) and
Role.allInstances()->forAll(r|r.senior->isEmpty() and r.junior->isEmpty())
```

The same constraint in relational logic:

```
(some u: User |
  (u + (u . ^PredSuccUser)) = User &&
  #(((ActiveUser . (u + (u . PredSuccUser))) . ActiveAccess) . AccessAction) = 2 &&
  no RoleHierarchy)
```

OCL2Kodkod returns the following complete and valid object diagram:



It results from the following USE command sequence which is automatically generated by OCL2Kodkod:

```
reset
!create user1:User
!create user2:User
!create session1:Session
!create session2:Session
!create clerk:Role
!create supervisor:Role
!create prepare:Action
!create approve:Action
!create cheque:Resource
!create access1:Access
!create access2:Access
!create snapshot1:Snapshot
!create snapshot2:Snapshot
!create p1:Permission between(prepare, cheque)
!create p2:Permission between(approve, cheque)
!create dsod:MutuallyExclusive between(clerk, supervisor)
!set dsod.wrtActiveRoles := true
!set user2.maxSessions := 1
!set user1.maxSessions := 1
!set clerk.maxSeniors := 2
!insert (p1,  clerk) into PermissionAssignment
!insert (p2,  supervisor) into PermissionAssignment
!insert (user1,  clerk) into UserAssignment
!insert (user1,  supervisor) into UserAssignment
!insert (user2,  clerk) into UserAssignment
!insert (session1,  user2) into ActiveUser
```

```
!insert (session2,  user1) into ActiveUser
!insert (session1,  clerk) into ActiveRoles
!insert (session2,  supervisor) into ActiveRoles
!insert (session1,  access2) into ActiveAccess
!insert (session2,  access1) into ActiveAccess
!insert (access1,  approve) into AccessAction
!insert (access2,  prepare) into AccessAction
!insert (access1,  cheque) into AccessResource
!insert (access2,  cheque) into AccessResource
!insert (snapshot2,  snapshot1) into PredSuccSnapshot
!insert (user2,  user1) into PredSuccUser
!insert (snapshot1,  user1) into SnapshotUser
!insert (snapshot2,  user2) into SnapshotUser
```

References

1. Martin Gogolla, Fabian Büttner, and Mark Richters. USE: A UML-Based Specification Environment for Validating UML and OCL. *Science of Computer Programming*, 69:27–34, 2007.