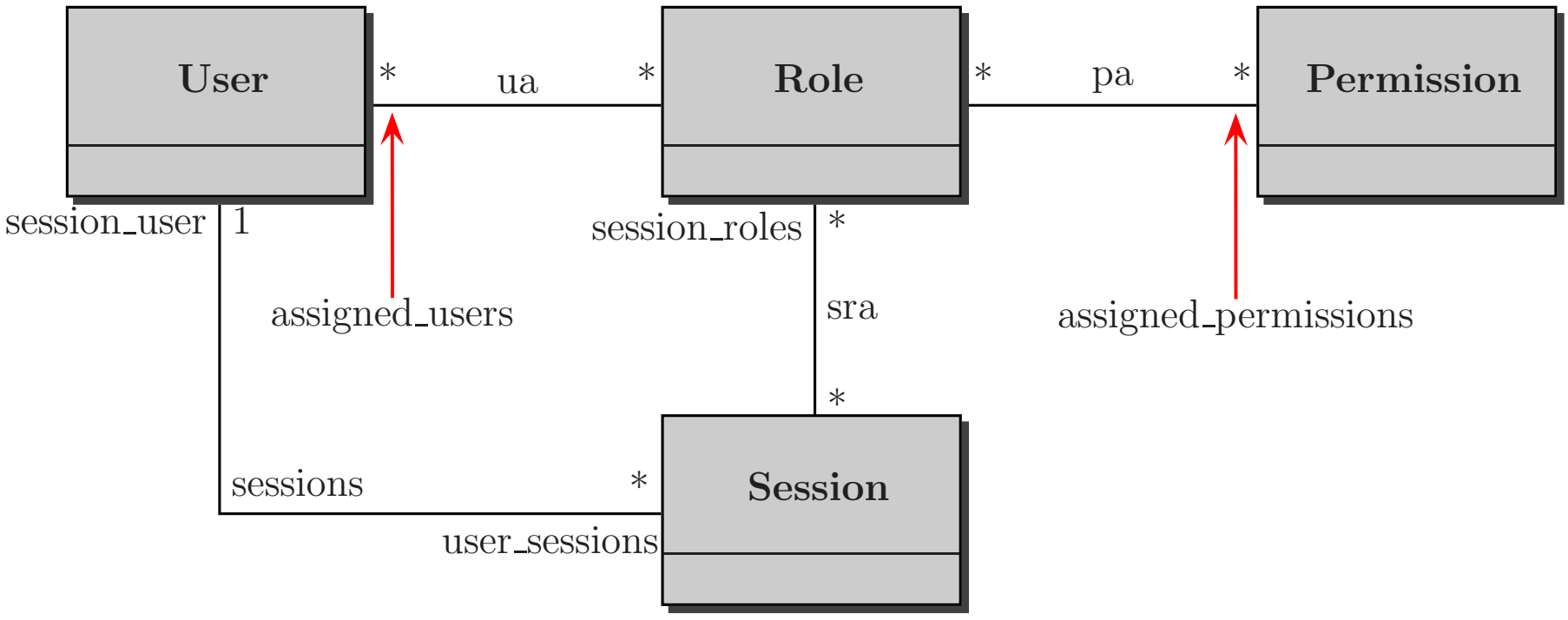
Formal Specification of Software

Role-based Access Control: An Example in OCL Formalisation

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

Class Diagram for the RBAC Core

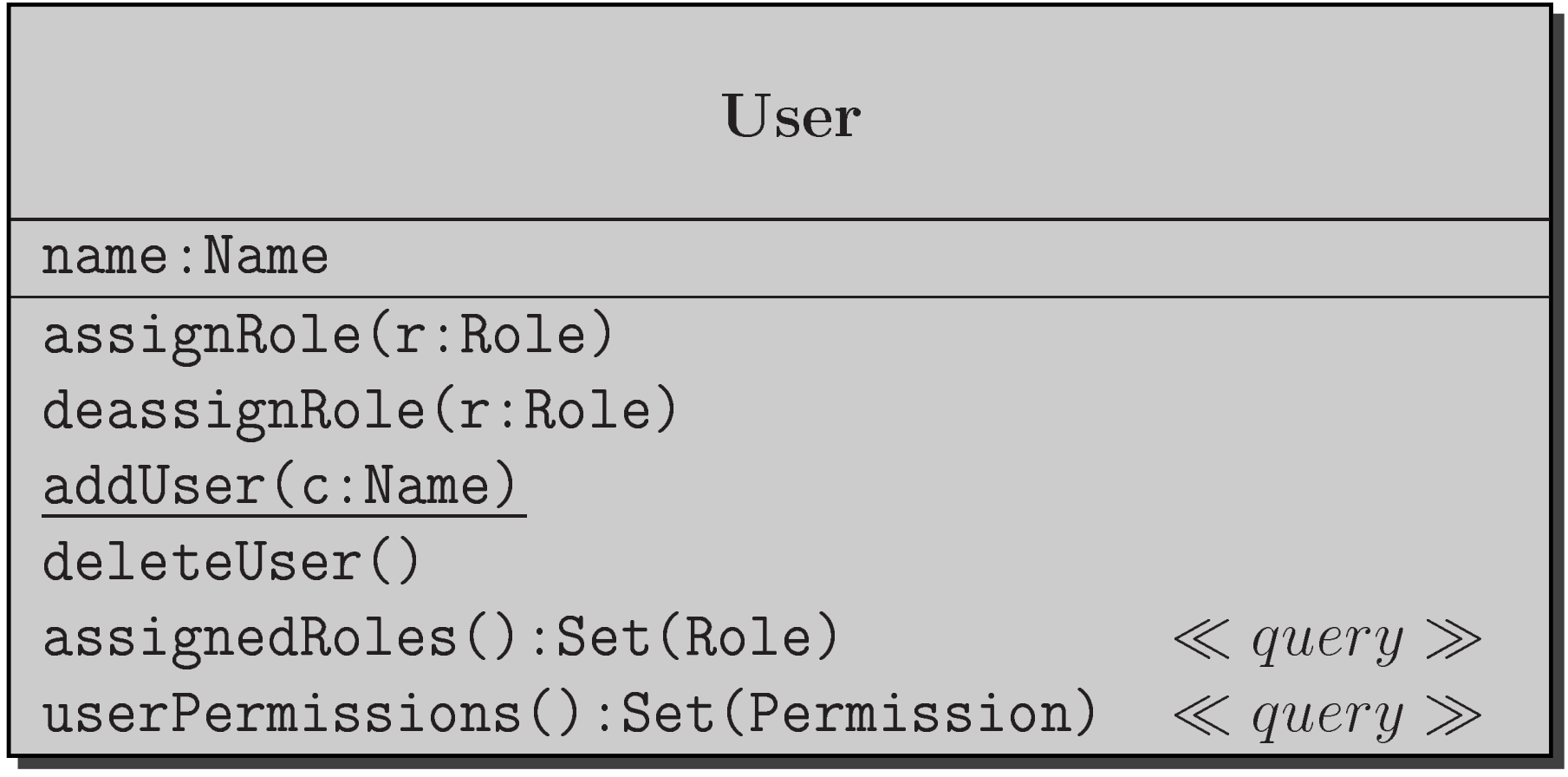


Top Level Constraint

context u:User

inv u.role->includesAll(u.user_sessions.session_roles)

The Class User



Constraints on Operations of Class User

assignRole

context u:User::assignRole(r:Role)

pre role->excludes(r)

post role = role@pre->including(r)

Constraints on Operations of Class User

assignRole

```
context  u:User::assignRole(r:Role)
  pre    role->excludes(r)
  post   role = role@pre->including(r)
```

deassignRole

```
context  u:User::deassignRole(r:Role)
  pre    role->includes(r)
  post   role = role@pre->excluding(r)    and
         u.user_sessions =
         u.user_sessions@pre->reject(
               s | s.session_roles->includes(r))
```

Constraints on Operations of Class User

addUser

context u:User::addUser(c:Name)

pre User.allInstances –> forAll(u_1 | u_1.name <> c)

post User.allInstances –> exists(u_1 |
User.allInstances@pre –> excludes(u_1) and
u_1.name = c and
User.allInstances = User.allInstances@pre –> including(u_1))

Constraints on Operations of Class User

addUser

context u:User::addUser(c:Name)

pre User.allInstances –> forAll(u_1 | u_1.name <> c)

post User.allInstances –> exists(u_1 |
User.allInstances@pre –> excludes(u_1) and
u_1.name = c and
User.allInstances = User.allInstances@pre –> including(u_1))

deleteUser

context u:User::deleteUser()

pre true

post User.allInstances = User.allInstances@pre –> excluding(u) and
Session.allInstances = Session.allInstances@pre –>
reject(s | u.user_sessions –> includes(s))

The Operation oclIsNew

context u:User::addUser(c:Name)

pre User.allInstances –> forAll(u_1 | u_1.name <> c)

post User.allInstances –> exists(u_1 |
User.allInstances@pre –> excludes(u_1) and
u_1.name = c and
User.allInstances = User.allInstances@pre –> including(u_1))

The Operation oclIsNew

context u:User::addUser(c:Name)

pre User.allInstances –> forAll(u_1 | u_1.name <> c)

post User.allInstances –> exists(u_1 |
User.allInstances@pre –> excludes(u_1) and
u_1.name = c and
User.allInstances = User.allInstances@pre –> including(u_1))

is equivalent to

context u:User::addUser(c:Name)

pre User.allInstances –> forAll(u_1 | u_1.name <> c)

post User.allInstances –> exists(u_1 |
u_1.oclIsNew and
u_1.name = c and
User.allInstances = User.allInstances@pre –> including(u_1))

Constraints on Operations of Class User

assignedRoles

context **u:User::assignedRoles()**

pre **true**

post **result = u.role**

Constraints on Operations of Class User

assignedRoles

context u:User::assignedRoles()

pre true

post result = u.role

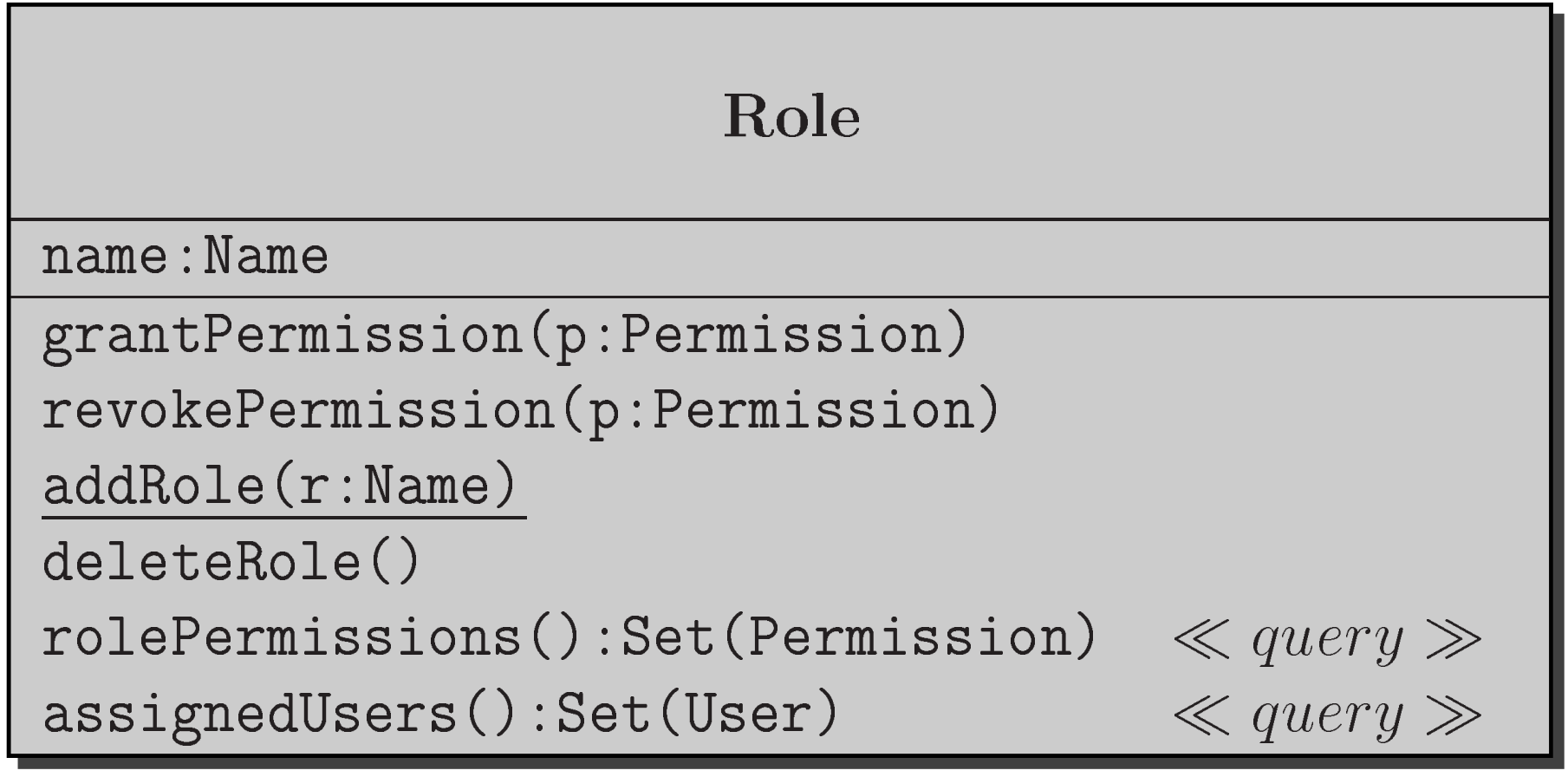
userPermissions

context u:User::userPermissions()

pre true

post result = u.role.assigned_permissions->asSet()

The Class Role



Constraints on Operations of Class Role

grantPermission

```
context  r:Role::grantPermission(p:Permission)
  pre    true
  post   r.assigned_permissions =
           r.assigned_permissions@pre->including(p)
```

Constraints on Operations of Class Role

grantPermission

context r:Role::grantPermission(p:Permission)

pre true

post r.assigned_permissions =
r.assigned_permissions@pre->including(p)

revokePermission

context r:Role::revokePermission(p:Permission)

pre r.assigned_permissions->includes(p)

post r.assigned_permissions =
r.assigned_permissions@pre->excluding(p)

Constraints on Operations of Class Role

addRole

context r:Role::addRole(c:Name)

pre Role.allInstances –> forAll(r | r.name <> c)

post Role.allInstances –> exists(r_1 |
r$_1$.oclIsNew and
r$_1$.name = c and
Role.allInstances = Role.allInstances@pre –> including(r$_1$))

Constraints on Operations of Class Role

addRole

context r:Role::addRole(c:Name)

pre Role.allInstances –> forAll(r | r.name <> c)

post Role.allInstances –> exists(r_1 |
r_1.oclIsNew and
r_1.name = c and
Role.allInstances = Role.allInstances@pre –> including(r_1))

deleteRole

context r:Role::deleteRole()

pre true

post Role.allInstances = Role.allInstances@pre –> excluding(r) and
Session.allInstances = Session.allInstances@pre –>
reject(s | s.session_roles –> includes(r))

Constraints on Operations of Class Role

rolePermissions

context r:Role::rolePermissions()
pre true
post result = r.assigned_permissions

Constraints on Operations of Class Role

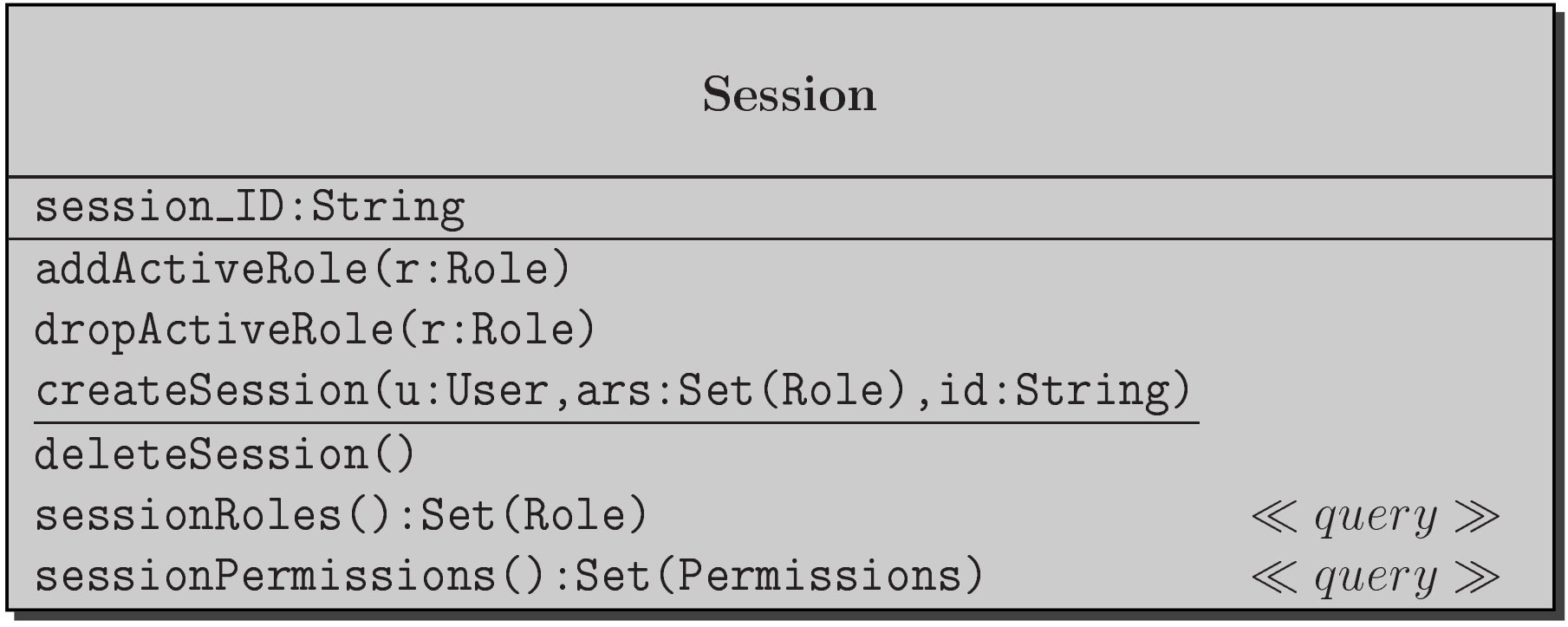
rolePermissions

context r:Role::rolePermissions()
  pre true
  post result = r.assigned_permissions

assignedUsers

context r:Role::assignedUsers()
  pre true
  post result = r.assigned_users

The Class Session



Constraints on Operations of Class Session

addActiveRole

context s:Session::addActiveRole(r:Role)

pre r.assigned_users –> includes(s.session_user) and
s.session_roles –> excludes(r)

post s.session_roles =
s.session_roles@pre –> including(r)

Constraints on Operations of Class Session

addActiveRole

context s:Session::addActiveRole(r:Role)

pre r.assigned_users->includes(s.session_user) and
s.session_roles->excludes(r)

post s.session_roles =
s.session_roles@pre->including(r)

dropActiveRole

context s:Session::dropActiveRole(r:Role)

pre s.session_roles->includes(r)

post s.session_roles =
s.session_roles@pre->excluding(r)

Constraints on Operations of Class Session

createSession

context s:Session::
createSession(u:User,ars:Set(Role),id:String)

pre Session.allInstances->forAll(s | s.session_ID <> id) and
u.role->includesAll(ars)

post u.user_sessions->exists(s_1 |
s_1.oclIsNew and
s_1.session_ID = id and
s_1.session_roles = ars
u.user_sessions =
u.user_sessions@pre->including(s_1) and
)

Constraints on Operations of Class Session

deleteSession

context **s:Session::deleteSession()**

pre **true**

post **Session.allInstances =**
Session.allInstances@pre->excluding(s)

Constraints on Operations of Class Session

sessionRoles

context **s:Session::sessionRoles()**

pre **true**

post **result = s.session_roles**

Constraints on Operations of Class Session

sessionRoles

context s:Session::sessionRoles()

pre true

post result = s.session_roles

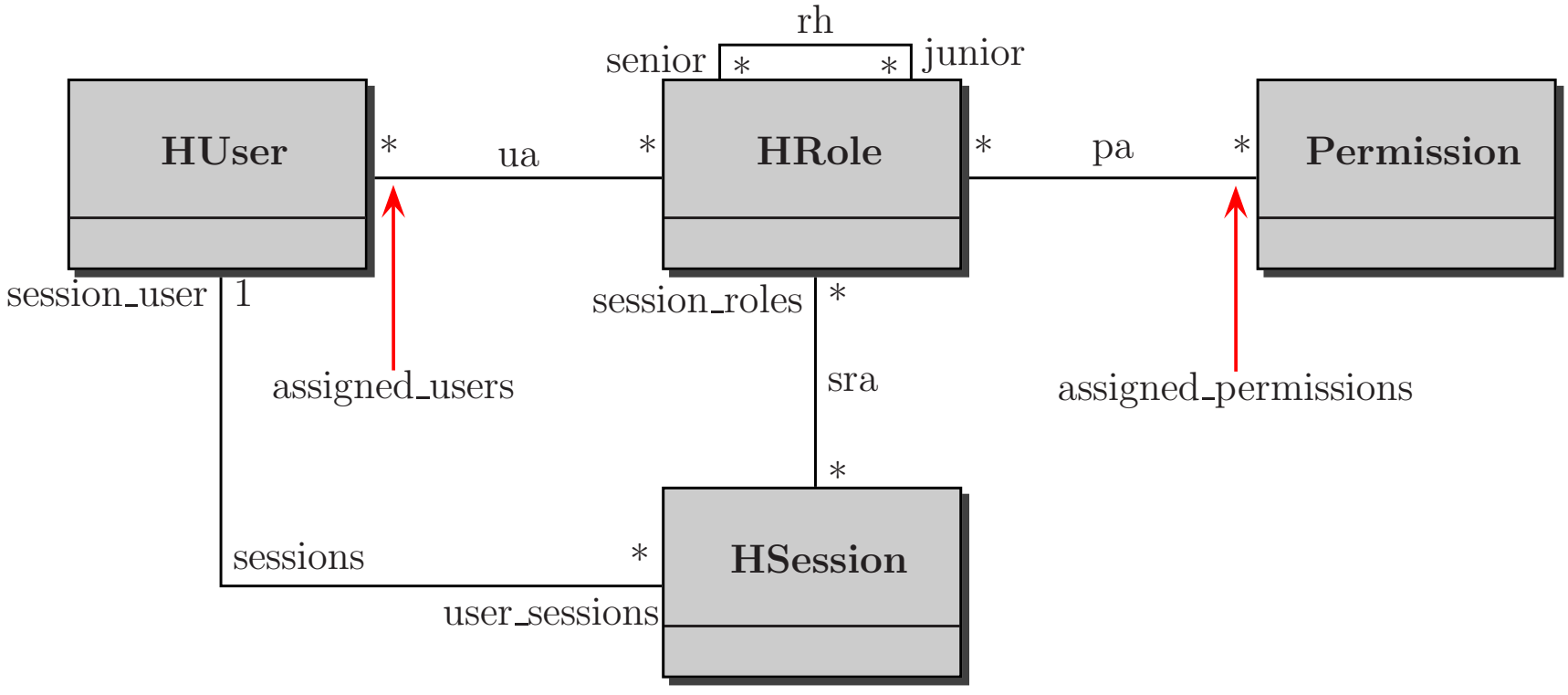
sessionPermissions

context s:Session::sessionPermissions()

pre true

post result = s.session_roles.assigned_permissions –> asSet()

Class Diagram for RBAC with Hierarchy



Auxilliary Definitions

senior$^+$

r.senior$^+$ abbreviation for

```
HRole.allInstances ->
  iterate(r1 ; y:Set(HRole) = r.senior | y -> union(y.senior))
```

Auxilliary Definitions

senior$^+$

r.senior$^+$ abbreviation for

```
HRole.allInstances ->
  iterate(r1 ; y:Set(HRole) = r.senior | y -> union(y.senior))
```

senior*

r.senior* abbreviation for

```
HRole.allInstances ->
  iterate(r1 ; y:Set(HRole) = { r } | y -> union(y.senior))
```

Auxilliary Definitions

senior$^+$

r.senior$^+$ abbreviation for

```
HRole.allInstances ->
  iterate(r1 ; y:Set(HRole) = r.senior | y -> union(y.senior))
```

senior*

r.senior* abbreviation for

```
HRole.allInstances ->
  iterate(r1 ; y:Set(HRole) = { r } | y -> union(y.senior))
```

junior$^+$

junior*

General Role Hierachies

No skipping of hierarchy levels, no cycles

```
context  r:HRole

  inv  GeneralRH :
    not HRole.allInstances ->exists(r1,r2 |
          r.senior ->includes(r1) and
          r.senior ->includes(r2) and
          r1.senior+ ->includes(r2))
    )  and
    r.senior+ ->excludes(r)
```

GeneralRH is the name of the invariant

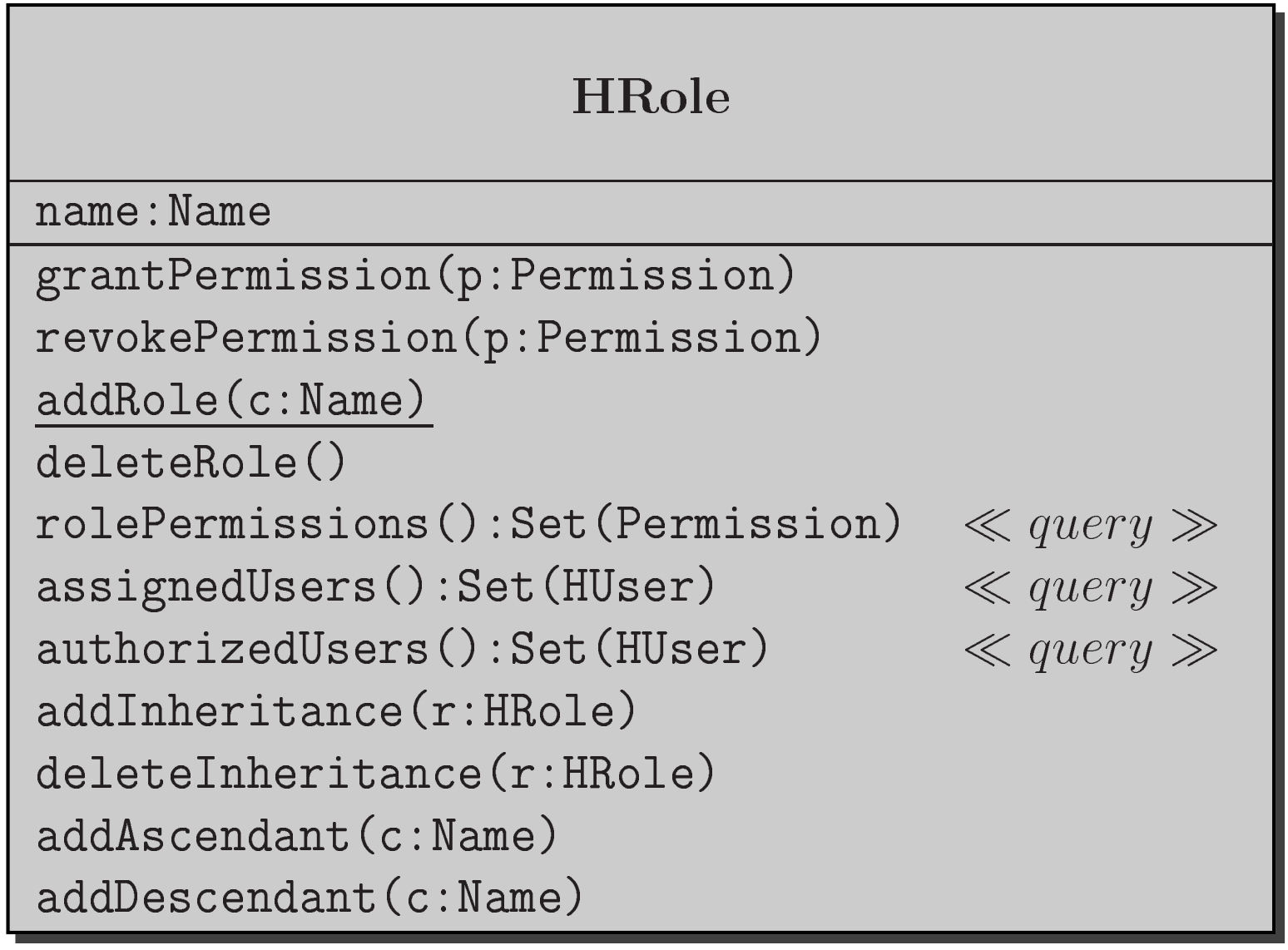
Limited Role Hierachies

Hierarchy a tree structure

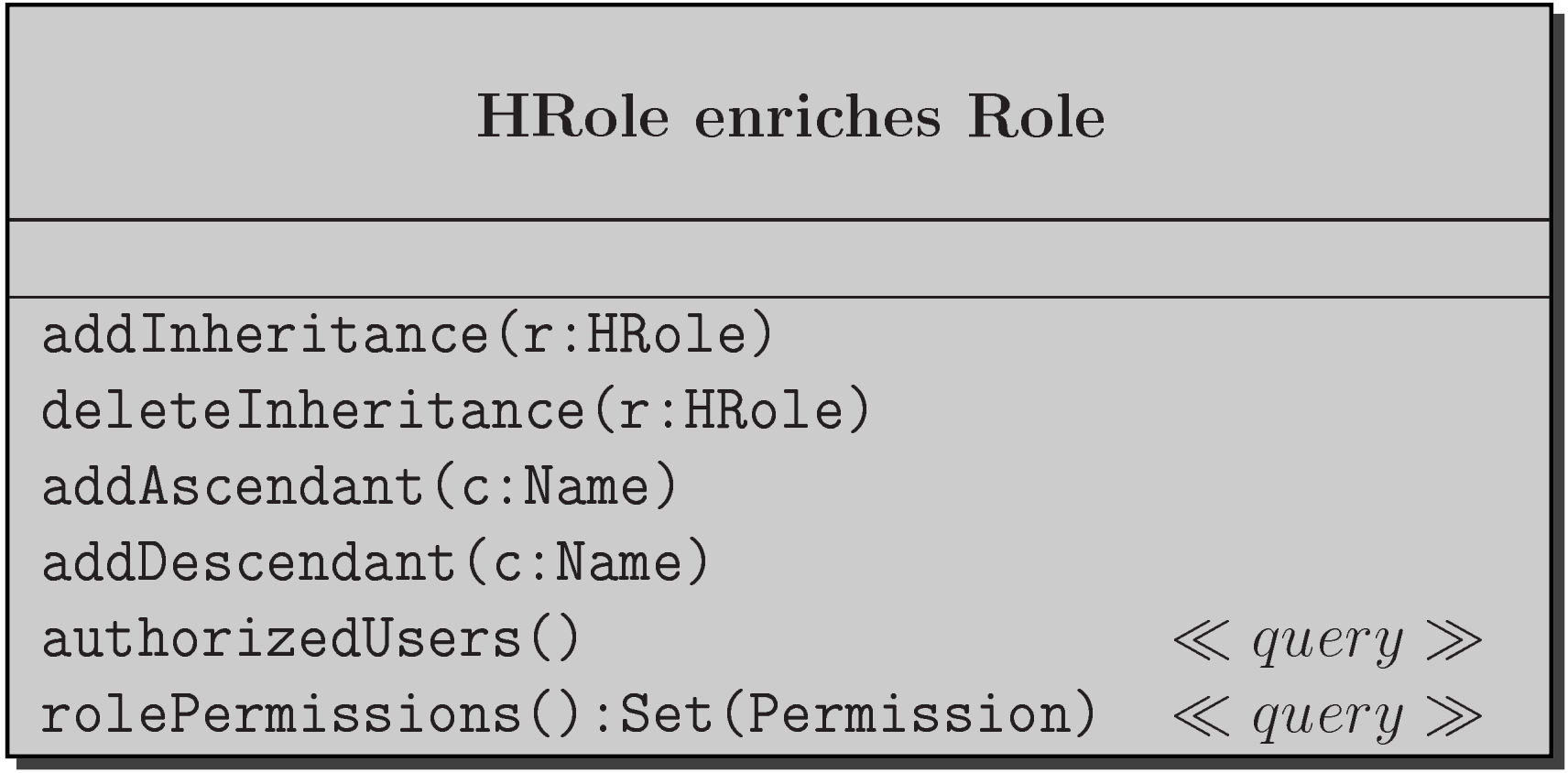
context r,r_1,r_2:HRole

inv LimitedRH :
GeneralRH and
(r.senior –> includes(r_1) and r.senior –> includes(r_2))
implies r_1 = r_2

Class HRole


HRole
name:Name
grantPermission(p:Permission)
revokePermission(p:Permission)
addRole(c:Name)
deleteRole()
rolePermissions():Set(Permission) ≪ query ≫
assignedUsers():Set(HUser) ≪ query ≫
authorizedUsers():Set(HUser) ≪ query ≫
addInheritance(r:HRole)
deleteInheritance(r:HRole)
addAscendant(c:Name)
addDescendant(c:Name)

Class HRole (Short version)



Constraints on Operations of Class HRole

addInheritance

context r:HRole::addInheritance(r_1:HRole)

pre not(r.senior* –> includes(r_1)) and
not(r_1.senior* –> includes(r))

post r.senior –> includes(r_1) and
HRole.allInstances –> forAll(r_2,r_3 |
((r_2 <> r or r_3 <> r_1) implies
(r_2.senior –> includes(r_3) iff
r_2.senior@pre –> includes(r_3))))

A iff B is an abbreviation for (A implies B) and (B implies A)

Constraints on Operations of Class HRole

deleteInheritance

context r:HRole::deleteInheritance(r_1:HRole)

pre r.senior->includes(r_1)

post r.senior->excludes(r_1) and
HRole.allInstances->forAll(r_2,r_3 |
((r_2 <> r or r_3 <> r_1) implies
(r_2.senior->includes(r_3) iff
r_2.senior@pre->includes(r_3))))

Constraints on Operations of Class HRole

addAscendant (combines addRole and addInheritance)

context r:HRole::addAscendant(c:Name)

pre HRole->forAll(r | r.name <> c)

post HRole.allInstances->exists(r_1 |
r_1.oclIsNew() and
r_1.name = c and
r.senior->includes(r_1) and
HRole.allInstances = HRole.allInstances@pre including(r_1)
)

Constraints on Operations of Class HRole

addDescendant **(combines addRole and addInheritance)**

```
context  r:HRole::addDescendant(c:Name)
  pre    HRole->forAll(r | r.name <> c)
  post   HRole.allInstances->exists( r1 |
           r1.oclIsNew() and
           r1.name = c and
           r1.senior->includes(r) and
           HRole.allInstances = HRole.allInstances@pre including(r1)
  )
```

Constraints on Operations of Class HRole

authorizedUsers

context r:HRole::authorizedUsers():Set(HUser)

pre true

post result =
r.senior* –> collect(r_1 | r_1.assigned_users) –> asSet()

Constraints on Operations of Class HRole

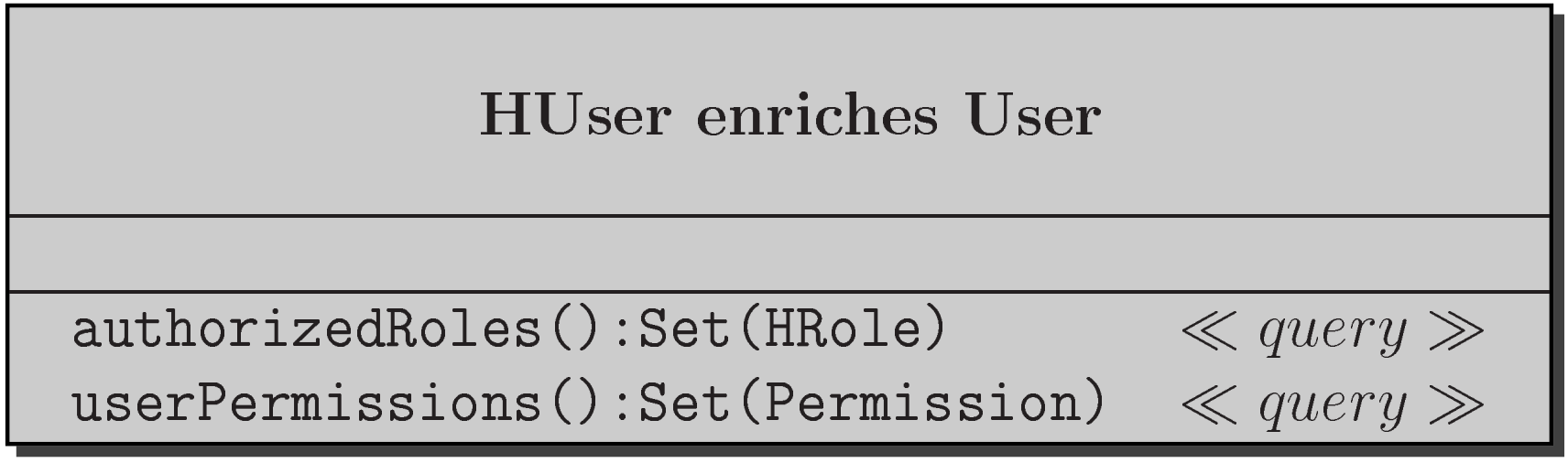
Redefining rolePermissions

context r:HRole::rolePermissions()

pre true

post result = r.junior* –> collect(r_1 | r_1.assigned_permissions)

Class HUser


HUser enriches User
authorizedRoles():Set(HRole) ≪ query ≫
userPermissions():Set(Permission) ≪ query ≫

Constraints on New Operations of Class HUser

authorizedRoles

context u:HUser::authorizedRoles():Set(HRole)

pre true

post result =

u.role –> collect(r | r.junior*) –> asSet()

Constraints on Operations of Class HUser

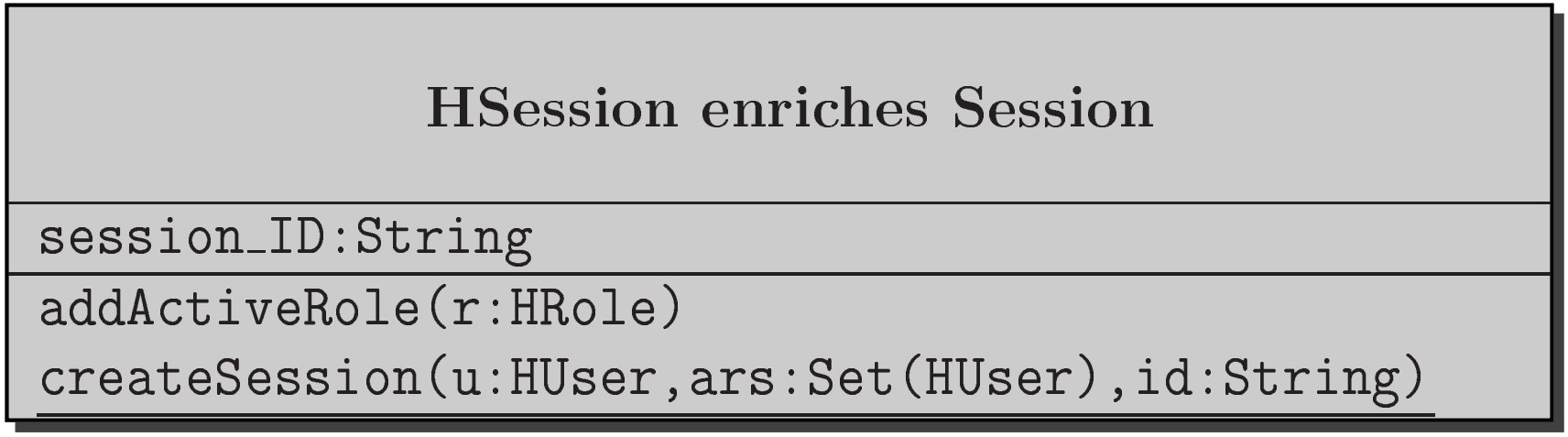
Redefining userPermissions

context u:HUser::userPermissions():Set(Permission)

pre true

post result =
u.authorizedRoles()->
collect(r_1 | r_1.assigned_permissions)->asSet()

Class HSession



Modified Constraints on Class HSession

addActiveRole

context s:HSession::addActiveRole(r:HRole)

pre r.authorizedUsers() –> includes(s.session_user) and
s.session_roles –> excludes(r)

post s.session_roles =
s.session_roles@pre –> including(r)

Modified Constraints on Class HSession

createSession

```
context  s:HSession::
    createSession(u:HUser,ars:Set(HRole),id:String)

    pre   HSessions.allInstances->forAll(s | s.session_ID <> id)   and
          u.authorizedRoles()->includesAll(ars)

    post  u.user_sessions->exists( s1 |
                s1.oclIsNew and
                s1.session_ID = id and
                u.user_sessions =
                      u.user_sessions@pre->including(s1) and
                s.session_roles = ars
          )
```

Derived Invariants

User inheritance relation

context r_1, r_2:HRole

inv UserInheritance:

r_1.senior* –> includes(r_2) implies

r_1.authorizedUsers() –>

includesAll(r_2.authorizedUsers())

Derived Invariants

User inheritance relation

context r_1, r_2:HRole

inv UserInheritance:

r_1.senior* –> includes(r_2) implies

r_1.authorizedUsers() –>

includesAll(r_2.authorizedUsers())

Permission inheritance relation

context r_1, r_2:HRole

inv PermissionInheritance:

r_1.senior* –> includes(r_2) implies

r_2.rolePermissions() –>

includesAll(r_1.rolePermissions())