```
THEORY Relation
  IMPORT THEORY PROJECTS
    /SimpleDEq THEORIES /SimpleDEq/Ring.dtf|org.eventb.theory.core.deployedTheoryRoot#Ring
  TYPE PARAMETERS S
  OPERATORS
    reflexive <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        ∀ x · x ∈ S ⇒ ((x ↦ x) ∈ rel)
    antireflexive <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        ∀ x · x ∈ S ⇒ (x ↦ x ∉ rel)
    symetrical <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        ∀ x,y · x ∈ S ∧ y ∈ S ⇒ (
          (x ↦ y ∈ rel) ⇒ (y ↦ x ∈ rel)
        )
    asymetrical <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        ∀ x,y · x ∈ S ∧ y ∈ S ⇒ (
          (x ↦ y ∈ rel) ⇒ (y ↦ x ∉ rel)
        )
    antisymetrical <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        ∀ x,y · x ∈ S ∧ y ∈ S ⇒ (
          (x ↦ y ∈ rel) ∧ (y ↦ x ∈ rel) ⇒ x = y
        )
    transitive <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        ∀ x, y, z · x ∈ S ∧ y ∈ S ∧ z ∈ S ⇒ (
          (x ↦ y ∈ rel) ∧ (y ↦ z ∈ rel) ⇒ (x ↦ z ∈ rel)
        )
    total <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        ∀ x, y · x ∈ S ∧ y ∈ S ⇒ ((x ↦ y ∈ rel) ∨ (y ↦ x ∈ rel))
    equivalence <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        reflexive(rel) ∧ symetrical(rel) ∧ transitive(rel)
    order <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        reflexive(rel) ∧ transitive(rel) ∧ antisymetrical(rel)
    strict <expression> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        { x ↦ y | x ↦ y ∈ rel ∧ x ≠ y }
    wellFounded <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        ∀ X · X ⊆ S ∧ X ≠ ∅ ⇒ (
          ∃ m · m ∈ X ∧ (∀ x · x ∈ X ⇒ (x ↦ m ∉ rel))
        )
    wellPartialOrder <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        order(rel) ∧ wellFounded(strict(rel))
    wellOrder <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        order(rel) ∧ total(rel) ∧ wellFounded(strict(rel))
    covers <predicate> (rel: S ↔ S,a: S,b: S)
      well−definedness S ≠ ∅ ∧ order(rel)
      direct definition
        (a ↦ b ∈ rel) ∧ a ≠ b ∧ (∀ c · c ∈ S ∧ (a ↦ c ∈ rel) ∧ (c ↦ b ∈ rel) ⇒ ((c = a) ∨ (b = a)))
    compose <expression> (rel1: S ↔ S,rel2: S ↔ S)
      well−definedness S ≠ ∅
```

```
      direct definition
        { x, z · x ∈ S ∧ z ∈ S ∧ (∃ y · y ∈ S ⇒ ((x ↦ y ∈ rel1) ∧ (y ↦ z ∈ rel2))) | x ↦ z }
    converse <expression> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        { x, y · x ∈ S ∧ y ∈ S ∧ y ↦ x ∈ rel | x ↦ y }
    complement <expression> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        { x,y · x ∈ S ∧ y ∈ S ∧ x ↦ y ∉ rel | x ↦ y }
    equality <expression> ()
      well−definedness S ≠ ∅
      direct definition
        { x ↦ y | x ∈ S ∧ y ∈ S ∧ x = y }
    preorder <predicate> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        reflexive(rel) ∧ transitive(rel)
    equivalenceClass <expression> (rel: S ↔ S,x: S)
      well−definedness S ≠ ∅ ∧ equivalence(rel)
      direct definition
        { y · y ∈ S ∧ (x ↦ y) ∈ rel | y }
    leftGeneralized <expression> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        { x,P · x ∈ S ∧ P ∈ ℙ(S) ∧ P ≠ ∅ ∧ (∀ y · y ∈ P ⇒ (x ↦ y ∈ rel)) | x ↦ P }
    rightGeneralized <expression> (rel: S ↔ S)
      well−definedness S ≠ ∅
      direct definition
        { P,x · P ∈ ℙ(S) ∧ P ≠ ∅ ∧ x ∈ S ∧ (∀ y · y ∈ P ⇒ (y ↦ x ∈ rel)) | P ↦ x }
    upperBound <predicate> (ord: S ↔ S,T: ℙ(S),B: S)
      direct definition
        ∀ t · t ∈ T ⇒ t ↦ B ∈ ord
    lowerBound <predicate> (ord: S ↔ S,T: ℙ(S),B: S)
      direct definition
        ∀ t · t ∈ T ⇒ B ↦ t ∈ ord
    bounds <predicate> (ord: S ↔ S,T: ℙ(S),m: S,M: S)
      direct definition
        lowerBound(ord,T,m) ∧ upperBound(ord,T,M)
    upperBounded <predicate> (ord: S ↔ S,T: ℙ(S))
      direct definition
        ∃ M · M ∈ S ∧ upperBound(ord,T,M)
    lowerBounded <predicate> (ord: S ↔ S,T: ℙ(S))
      direct definition
        ∃ m · m ∈ S ∧ lowerBound(ord,T,m)
    bounded <predicate> (ord: S ↔ S,T: ℙ(S))
      direct definition
        ∃ m, M · m ∈ S ∧ M ∈ S ∧ bounds(ord,T,m,M)
    supremum <predicate> (ord: S ↔ S,T: ℙ(S),M: S)
      direct definition
        upperBound(ord,T,M) ∧
        (∀ m · m ∈ S ∧ upperBound(ord,T,m) ⇒ M ↦ m ∈ ord)
    infimum <predicate> (ord: S ↔ S,T: ℙ(S),m: S)
      direct definition
        lowerBound(ord,T,m) ∧
        (∀ M · M ∈ S ∧ lowerBound(ord,T,M) ⇒ M ↦ m ∈ ord)
    maximal <predicate> (ord: S ↔ S,T: ℙ(S),M: T)
      direct definition
        ∀ x · x ∈ T ∧ M ↦ x ∈ ord ⇒ x = M
    minimal <predicate> (ord: S ↔ S,T: ℙ(S),M: T)
      direct definition
        ∀ x · x ∈ T ∧ x ↦ M ∈ ord ⇒ x = M
    maximum <predicate> (ord: S ↔ S,T: ℙ(S),M: S)
      direct definition
        (M ∈ T) ∧ (∀ x · x ∈ T ⇒ x ↦ M ∈ ord)
    minimum <predicate> (ord: S ↔ S,T: ℙ(S),M: S)
      direct definition
        (M ∈ T) ∧ (∀ x · x ∈ T ⇒ M ↦ x ∈ ord)
    hasMaximum <predicate> (ord: S ↔ S,T: ℙ(S))
      direct definition
        ∃ M · M ∈ T ∧ maximum(ord,T,M)
    hasMinimum <predicate> (ord: S ↔ S,T: ℙ(S))
      direct definition
        ∃ m · m ∈ T ∧ minimum(ord,T,m)
```

```
    monoidCompatible <predicate> (op: (S × S) → S,e: S,rel: S ↔ S)
      well−definedness S ≠ ∅ ∧ Monoid(op,e) ∧ order(rel)
      direct definition
        ∀ x, y, z · x ∈ S ∧ y ∈ S ∧ z ∈ S ∧ (x ↦ y ∈ rel) ⇒ (
          ((op(x ↦ z) ↦ op(y ↦ z)) ∈ rel) ∧
          ((op(z ↦ x) ↦ op(z ↦ y)) ∈ rel)
        )
    ringCompatible <predicate> (oplus: (S × S) → S,otimes: (S × S) → S,azero: S,unit: S,rel: S ↔ S)
      well−definedness S ≠ ∅ ∧ Ring(oplus,otimes,azero,unit) ∧ order(rel)
      direct definition
        monoidCompatible(oplus,azero,rel) ∧ (
          ∀ x, y · x ∈ S ∧ y ∈ S ∧ (azero ↦ x ∈ rel) ∧ (azero ↦ y ∈ rel) ⇒
            ((azero ↦ otimes(x ↦ y) ∈ rel) ∧ (azero ↦ otimes(y ↦ x) ∈ rel))
        )
  AXIOMATIC DEFINITIONS  operations:
    OPERATORS
      Gmax <expression> (ord: S ↔ S,T: ℙ(S)) : S
      Gmin <expression> (ord: S ↔ S,T: ℙ(S)) : S
      Gsup <expression> (ord: S ↔ S,T: ℙ(S)) : S
      Ginf <expression> (ord: S ↔ S,T: ℙ(S)) : S
    AXIOMS
      GmaxDef:
        ∀ ord,T · ord ∈ S ↔ S ∧ T ⊆ S ∧ hasMaximum(ord,T) ⇒ (maximum(ord,T,Gmax(ord,T)))
      GminDef:
        ∀ ord,T · ord ∈ S ↔ S ∧ T ⊆ S ∧ hasMinimum(ord,T) ⇒ (minimum(ord,T,Gmin(ord,T)))
      GsupDef:
        ∀ ord,T · ord ∈ S ↔ S ∧ T ⊆ S ∧ upperBounded(ord,T) ⇒ (supremum(ord,T,Gsup(ord,T)))
      GinfDef:
        ∀ ord,T · ord ∈ S ↔ S ∧ T ⊆ S ∧ lowerBounded(ord,T) ⇒ (infimum(ord,T,Ginf(ord,T)))
  THEOREMS
    converseDomain:
      ∀ rel · rel ∈ (S ↔ S) ⇒ (dom(converse(rel)) = ran(rel))
    converseRange:
      ∀ rel · rel ∈ (S ↔ S) ⇒ (ran(converse(rel)) = dom(rel))
    converseInvolutive:
      ∀ rel · rel ∈ (S ↔ S) ⇒ (converse(converse(rel)) = rel)
    converseSymetry:
      ∀ rel · rel ∈ (S ↔ S) ∧ symetrical(rel) ⇒ symetrical(converse(rel))
    converseReflexivity:
      ∀ rel · rel ∈ (S ↔ S) ∧ reflexive(rel) ⇒ reflexive(converse(rel))
    converseAntireflexivity:
      ∀ rel · rel ∈ (S ↔ S) ∧ antireflexive(rel) ⇒ antireflexive(converse(rel))
    complementInvolutive:
      ∀ rel · rel ∈ (S ↔ S) ⇒ (complement(complement(rel)) = rel)
    complementConverse:
      ∀ rel · rel ∈ (S ↔ S) ⇒ (complement(converse(rel)) = converse(complement(rel)))
    complementSymetry:
      ∀ rel · rel ∈ (S ↔ S) ∧ symetrical(rel) ⇒ symetrical(complement(rel))
    complementReflexivity:
      ∀ rel · rel ∈ (S ↔ S) ∧ reflexive(rel) ⇒ antireflexive(complement(rel))
    complementAntireflexivity:
      ∀ rel · rel ∈ (S ↔ S) ∧ antireflexive(rel) ⇒ reflexive(complement(rel))
    totalRelationsReflexivity:
      ∀ rel · rel ∈ (S ↔ S) ∧ total(rel) ⇒ reflexive(rel)
    equivalenceClassEquity:
      ∀ rel,x ,y · rel ∈ (S ↔ S) ∧ equivalence(rel) ∧ x ∈ S ∧ y ∈ S ⇒ (
        ((x ↦ y) ∈ rel) ⇔ (equivalenceClass(rel, x) = equivalenceClass(rel, y))
      )
    equivalenceClassNotEmpty:
      ∀ rel, x · rel ∈ (S ↔ S) ∧ equivalence(rel) ∧ x ∈ S ⇒
        (equivalenceClass(rel,x) ≠ ∅)
    equivalenceClassCover:
      ∀ rel · rel ∈ (S ↔ S) ∧ equivalence(rel) ⇒ (
        (⋃ equivalenceClass(rel,x) | x ∈ S) = S
      )
    equivalenceClassDisjoint:
      ∀ rel, x, y · rel ∈ (S ↔ S) ∧ equivalence(rel) ∧ x ∈ S ∧ y ∈ S ∧ (x ↦ y ∉ rel) ⇒ (
        (equivalenceClass(rel,x) ∩ equivalenceClass(rel,y)) = ∅
      )
    supremumMaximal:
      ∀ ord, T, M · ord ∈ (S ↔ S) ∧ order(ord) ∧ T ⊆ S ∧ M ∈ S ⇒
        (supremum(ord,T,M) ∧ M ∈ T ⇒ maximal(ord,T,M))
    infimumMinimal:
      ∀ ord, T, M · ord ∈ (S ↔ S) ∧ order(ord) ∧ T ⊆ S ∧ M ∈ S ⇒
```

```
        (infimum(ord,T,M) ∧ M ∈ T ⇒ minimal(ord,T,M))
END
```