Algorithm 1 Conflict Detector

Require: P: set of policies, R: set of relationships

```
function CONFLICT DETECTOR (P,R)
  PLAYPROPAGATION (P,R)
  OWNERSHIPPROPAGATION (P,R)
  ORGHIERARCHYPROPAGATION (P,R)
  ROLEHIERARCHYPROPAGATION (P,R)
  OBJECTCOMPOSITIONPROPAGATION (P,R)
  conflict = false
  for all $p_1 \in$ do
    for all $p_2 \in P$ do
      if $(p_1.org = p_2.org) \wedge (p_1.a = p_2 .a) \wedge$
        $(p_1.ov = p_2.ov)$ then
          if intersect $(p_1, p_2)$ then
            conflict $\leftarrow$ ORTHOGONALROLECR $(p_1,p_2,P)$
      if $(p_1.org = p_2.org) \wedge (p_1.a = p_2 .a) \wedge$
        $(p_1.sr = p_2.sr)$ then
          if intersect $(p_1, p_2)$ then
            conflict $\leftarrow$ ORTHOGONALVIEWCR $(p_1,p_2,P)$
      if $(p_1.sr = p_2.sr) \wedge (p_1.a = p_2 .a) \wedge$
        $(p_1.ov = p_2.ov)$ then
          if intersect $(p_1, p_2)$ then
            conflict $\leftarrow$ ORTHOGONALORGCR $(p_1,p_2,P)$
      if $(p_1.org = p_2.org) \wedge (p_1.sr = p_2 .sr) \wedge$
        $(p_1.ov = p_2.ov)$ then
          if intersect $(p_1, p_2)$ then
            conflict $\leftarrow$ $p_1.a = p_2.a \vee$
              REFINEMENTCR $(p_1,p_2,P) \vee$
              COMPOSITIONCR $(p_1,p_2,P) \vee$
              ORTHOGONALCR $(p_1,p_2) \vee$
              DEPENDENCYCR $(p_1,p_2)$
  return Conflict
```

Algorithm 2 PlayPropagation

```
Require: P: set of policies, R: set of relationships
function PLAYPROPAGATION (P,R)
  for all $p \in P$ do
    if (p.sr.type = “role”) then
      for all $s \in$ R.PLAY (p.org,p.r) do
        $p_1 \leftarrow \langle$p.kp,p.org,s,p.a,p.ov,p.ac,p.dc$\rangle$
        P $\leftarrow$ INCLUDEPOLICIE ($p_1$,P)
return P
```

Algorithm 3 OwnershipPropagation

```
Require: P: set of policies, R: set of relationships
function OWNERSHIPPROPAGATION (P,R)
  for all $p \in P$ do
    if (p.sr.type = “$\varnothing$”) then
      for all $r \in$ R. OWNERSHIP (p.org) do
        $p_1 \leftarrow \langle$ p.kp,p.org,r,p.a,p.ov,p.ac,p.dc$\rangle$
        P $\leftarrow$ INCLUDEPOLICIE ($p_1$,P)
return P
```

Algorithm 4 OrghierarchyPropagation

```
Require: P: set of policies, R: set of relationships
function ORGHIERARCHYPROPAGATION (P,R)
  for all $p \in P$ do
    if (p.sr = “$\varnothing$”) then
      for all org $\in$ R. ORGHIERARCHY(p.org) do
        $p_1 \leftarrow \langle$ p.kp,p.org,org,p.a,p.ov,p.ac,p.dc$\rangle$
        P $\leftarrow$ INCLUDEPOLICIE ($p_1$,P)
return P
```

Algorithm 5 RoleHierarchyPropagation

```
Require: P: set of policies, R: set of relationships
function ROLEHIERARCHYPROPAGATION (P,R)
  for all $p \in P$ do
    if (p.sr.type = “role”) then
      for all $r \in$ R. ROLEHIERARCHY (p.org, p.r) do
        $p_1 \leftarrow \langle$ p.kp,p.org,r,p.a,p.ov,p.ac,p.dc$\rangle$
        P $\leftarrow$ INCLUDEPOLICIE ($p_1$,P)
return P
```

Algorithm 6 ObjectPropagation

Require: P: set of policies, R: set of relationships

```
function OBJECTPROPAGATION (P,R)
  for all $p \in P$ do
    if (p.vo.type = “view”) then
      for all $o \in$ R. OBJECTCOMPOSITION (p.vo) do
        $p_1 \leftarrow \langle$ p.kp,p.org,p.sr,p.a,o,p.ac,p.dc$\rangle$
        P $\leftarrow$ INCLUDEPOLICIE ($p_1$,P)
return  P
```

Algorithm 7 RefinementCR

Require: P: set of policies, p_1 and p2; two policies

```
function REFINEMENTCR ($p_1$,$p_2$,P)
    if (($p_1$.kp = “F”) $\wedge$ ($p_2$.kp = “O” $\vee$ “P”)) $\wedge$
        REFINEMENT ($p_1$.a, $p_2$.a)) then
        return true
    if (($p_1$.kp =  (“O” $\vee$ “P” )) $\wedge$ ($p_2$.kp = “F”) $\wedge$
        REFINEMENT ($p_1$.a,$p_2$.a)) then
        for all $a \in$ SUBACTIONSOF ($p_1$.a) do
            for all $p \in P$ do
                if ((p.a = a) $\wedge$
                    (p.kp$\neq$”F”)) then
                    return false
        return true
```

Algorithm 8 CompositionCR

Require: P: set of policies, p_1 and p_2; two policies

```
function COMPOSITIONCR($p_1$,$p_2$,P)
    if (($p_1$.kp = (“O” $\vee$ “P”)) $\wedge$ ($p_2$.kp =  “F”)) $\wedge$
      COMPOSITION($p_1$.a,$p_2$.a)) then
        return true
    if (($p_1$.kp = “F”) $\wedge$ ($p_2$.kp =  (“O” $\vee$ “P” ))) $\wedge$
      COMPOSITION($p_1$.a,$p_2$.a)) then
        for all $a \in$ PARTACTIONSOF($p_1$.a) do
            for all $p \in P$ do
            if ((p.a = a) $\wedge$ (p.kp$\neq$ (”O” $\wedge$ “P”))) then
                return false
        return true
    return false
```

Algorithm 9 OrthogonalCR

Require: p_1 and $p_{2:}$ two policies

```
function ORTHOGONALCR($p_1$,$p_2$)
    if (($p_1$.kp = “O”) $\wedge$ ($p_2$.kp = (“O” $\vee$ “P” ))) $\wedge$
        ORTHOGONAL($p_1$.a,$p_2$.a)) then
            return true
return false
```

Algorithm 10 DependencyCR

Require: p_1 and $p_{2:}$ two policies

```
function DEPENDENCYCR($p_1$,$p_2$)
    if (($p_1$.kp = “O” $\vee$ “P”)) $\wedge$ ($p_2$.kp = “F”) $\wedge$
        DEPENDENT($p_1$.a,$p_2$.a)) then
            return true
return false
```