**Algorithm 1** Conflict Detector

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

1: **function** CONFLICT DETECTOR (P,R)

2: PLAYPROPAGATION (P,R)

3: OWNERSHIPPROPAGATION (P,R)

4: ORGHIERARCHYPROPAGATION (P,R)

5: ROLEHIERARCHYPROPAGATION (P,R)

6: OBJECTCOMPOSITIONPROPAGATION (P,R)

7: conflict = false

8**: for all** p1 ∈ **do**

9: **for all** p2 ∈ P **do**

10: **if** (p1.org = p2.org) ∧ (p1.a = p2 .a) ∧

11: (p1.ov = p2.ov) **then**

12: if intersect (p1, p2) **then**

13: conflict ← ORTHOGONALROLECR (p1,p2,P)

14: **if** (p1.org = p2.org) ∧ (p1.a = p2 .a) ∧

15: (p1.sr = p2.sr) **then**

16: if intersect (p1, p2) **then**

17: conflict ← ORTHOGONALVIEWCR (p1,p2,P)

18: **if** (p1.sr = p2.sr) ∧ (p1.a = p2 .a) ∧

19: (p1.ov = p2.ov) **then**

20: if intersect (p1, p2) **then**

21: conflict ← ORTHOGONALORGCR (p1,p2,P)

22: **if** (p1.org = p2.org) ∧ (p1.sr = p2 .sr) ∧

23: (p1.ov = p2.ov) **then**

24: if intersect (p1, p2) **then**

25: conflict ← p1.a = p2.a ∨

26: REFINEMENTCR (p1,p2,P) ∨

27: COMPOSITIONCR (p1,p2,P) ∨

28: ORTHOGONALCR (p1,p2) ∨

29: DEPENDENCYCR (p1,p2)

30**: return** Conflict

**Algorithm 2** PlayPropagation

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

1: **function** PLAYPROPAGATION (P,R)

2: **for all** p∈ P **do**

3:  **if** (p.sr.type= “role”) **then**

4:  **for all** s∈ R.PLAY (p.org,p.r) **do**

5: p1 ← 〈p.kp,p.org,s,p.a,p.ov,p.ac,p.dc〉

6: P ← INCLUDEPOLICIE (p1,P)

7: **return** P

**Algorithm 3** OwnershipPropagation

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

1: **function** OWNERSHIPPROPAGATION (P,R)

2: **for all** p∈ P **do**

3:  **if** (p.sr.type= “∅”) **then**

4:  **for all** r∈ R. OWNERSHIP (p.org) **do**

5: p1 ← 〈 p.kp,p.org,r,p.a,p.ov,p.ac,p.dc〉

6: P ← INCLUDEPOLICIE (p1,P)

7: **return** P

**Algorithm 4** OrghierarchyPropagation

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

1: **function** ORGHIERARCHYPROPAGATION (P,R)

2: **for all** p∈ P **do**

3:  **if** (p.sr= “∅”) **then**

4:  **for all** org∈ R. ORGHIERARCHY(p.org) **do**

5: p1 ← 〈 p.kp,p.org,org,p.a,p.ov,p.ac,p.dc〉

6: P ← INCLUDEPOLICIE (p1,P)

7: **return** P

**Algorithm 5** RoleHierarchyPropagation

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

1: **function** ROLEHIERARCHYPROPAGATION (P,R)

2: **for all** p∈ P **do**

3:  **if** (p.sr.type= “role”) **then**

4:  **for all** r∈ R. ROLEHIERARCHY (p.org, p.r) **do**

5: p1 ← 〈 p.kp,p.org,r,p.a,p.ov,p.ac,p.dc〉

6: P ← INCLUDEPOLICIE (p1,P)

7: **return** P

**Algorithm 6** ObjectPropagation

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

1: **function** OBJECTPROPAGATION (P,R)

2: **for all** p∈ P **do**

3:  **if** (p.vo.type= “view”) **then**

4:  **for all** o∈ R. OBJECTCOMPOSITION (p.vo) **do**

5: p1 ← 〈 p.kp,p.org,p.sr,p.a,o,p.ac,p.dc〉

6: P ← INCLUDEPOLICIE (p1,P)

7: **return** P

**Algorithm 8** CompositionCR

**Require: P**: set of policies, p1 and p2; two policies

1: **function** COMPOSITIONCR(p1,p2,P)

2: **if (**(p1.kp = (“O” ∨ “P”)) ∧ (p2.kp= “F”)) ∧

3: COMPOSITION(p1.a,p2.a)) **then**

4: **return** *true*

5: **if (**(p1.kp = “F”) ∧ (p2.kp= (“O” ∨ “P” ))) ∧

6: COMPOSITION(p1.a,p2.a)) **then**

7: **for all** a∈ PARTACTIONSOF(p1.a) **do**

8: **for all** p∈ P **do**

9: **if** ((p.a = a) ∧ (p.kp≠ (”O” ∧ “P”))) **then**

10: **return** *false*

11: **return** *true*

12: **return** *false*

**Algorithm 7** RefinementCR

**Require: P**: set of policies, p1 and p2; two policies

1: **function** REFINEMENTCR (p1,p2,P)

2: **if (**(p1.kp= “F”) ∧ (p2.kp = “O” ∨ “P”)) ∧

3: REFINEMENT (p1.a, p2.a)) **then**

4: **return** *true*

5: **if (**(p1.kp= (“O” ∨ “P” )) ∧ (p2.kp = “F”) ∧

6: REFINEMENT (p1.a,p2.a)) **then**

7: **for all** a∈ SUBACTIONSOF (p1.a) **do**

8: **for all** p∈ P **do**

9: **if** ((p.a = a) ∧

10: (p.kp≠”F”)) **then**

11: **return** *false*

12: **return** *true*

13: **return** *false*

**Algorithm 10** DependencyCR

**Require:** p1 and p2: two policies

1: **function** DEPENDENCYCR(p1,p2)

2:  **if (**(p1.kp = “O” ∨ “P”)) ∧ (p2.kp= “F”) ∧

3: DEPENDENT(p1.a,p2.a)) **then**

4: **return** *true*

5: **return** *false*

**Algorithm 9** OrthogonalCR

**Require:** p1 and p2: two policies

1: **function** ORTHOGONALCR(p1,p2)

2: **if (**(p1.kp = “O”) ∧ (p2.kp= (“O” ∨ “P” ))) ∧

3: ORTHOGONAL(p1.a,p2.a)) **then**

4: **return** *true*

5: **return** *false*