Journal of Mobile Multimedia, Vol. 10, No. 1&2 (2014) 003-015
© Rinton Press

A PATH-CONSISTENCY BASED ALGORITHM FOR ANOMALY DETECTION OF SPATIAL CONSTRAINTS IN GEOXACML POLICIES

TUAN ANH DANG

Ho Chi Minh City University of Technology, VNUHCM

dangtuananh.dangtuananh@gmail.com

TRAN KHANH DANG

Ho Chi Minh City University of Technology, VNUHCM

khanh@cse.hcmut.edu.vn

Anomaly detection in GeoXACML policies supports policy designers in the policy definition process to save effort, minimize errors, improve performance. Currently, there is a lot of research focusing on anomaly detection in XACML from which GeoXACML is extended. However, no research directly solves anomaly detection problem in GeoXACML, especially in the spatial aspect. In this paper, we propose an algorithm based on the path-consistency algorithm to detect anomalies in spatial constraints of GeoXACML policies. In our approach, when a policy designer adds a new rule or updates an existing rule, an engine will automatically check if this rule is potentially conflicting or redundant to others. We also present a simple example in step-by-step to clarify how this algorithm works. Finally, to analyze the performance of this algorithm, we will consider its computational complexity.

Keywords. GeoXACML, anomaly detection, RCC8, path-consistency algorithm.

1. Introduction

Geographic information system (GIS) is more and more developed incessantly in both quantity and quality, and is applied in many fields such as: study of natural resources and environment, study of socio-economic condition, support for planning process, etc. Data in these systems maybe contain sensitive and/or confidential information, so ensuring access restrictions to these spatial data is very important. GeoXACML [1], a spatial extension to XACML Version 2.0 [2], is used for that purpose. It is an XML based policy language to express spatial access rights.

The structure of GeoXACML policies is similar to that of XACML policies as shown in figure 1. A rule element, the most basic block, describes an access rule in an organization. Each rule contains a target, a condition and an effect. The target of a rule checks whether an access request is applicable to the rule. Then, the condition determines that that request is able to receive the effect or not. The effect can get two values: Permit or Deny. If an access request satisfies both the target and the condition of a rule, the response will be specified by the effect element in that rule. Otherwise, the response is Not Applicable. A set of rules is combined in a policy with a target and a rule combining algorithm. In a similar way, a set of policies is contained in a policy set with a target and a policy combining algorithm. The target of a policy or a policy set defines a set of subjects, actions, resources and environments which the policy or policy set apply to. The rule and policy combining algorithm define how to decide the final effect if the effect of rules or policies that are applied to the access request are different. This case is called conflict in policies. To resolve conflicts, GeoXACML defines four different combining algorithms: Deny-Overrides, Permit-Overrides, First-Applicable and Only-One-Applicable [2] which decide the final result from the conflicting rules.

```
 <PolicySet PolicySetId="PS1"
PolicyCombiningAlgId="First-Applicable">
<Target>
<Policy PolicyId="P1"
    RuleCombiningAlgId="Permit-Overrides">
    <Target>
    <Rule RuleId="R1" Effect="Permit">
            <Target>
                    <Subjects><Subject>     Doctor
                            <Subject>    Surgeons
                    <Actions><Action>       Read
                    <Resources><Resource>PatientRecord
            <Condition>
            8:00≤Time ∧ Time≤17:00
            ∧ within(Location, Polygon(0 0,10 0,10 10,0 10))
```

Figure 1 An example of GeoXACML policy (closing tags are omitted).

Another type of anomaly in GeoXACML policies is redundancy. Two roles are redundant if there is an access request can be applied by both these rules with a same effect. Because response time of an access request is largely dependent on the number of rules in the policy repository, redundancies may affect the performance of evaluation process. Therefore, they are also considered as anomalies and should be resolved to optimize the policy evaluation.

Unfortunately, GeoXACML does not have any system mechanism to detect anomalies in polices. The selection of combining algorithm solely replies on the knowledge and experience of policy designers without any supporting conflict information from the system, so the correctness of this selection for resolving policy conflicts will not be guaranteed.

Recently, anomaly detection in XACML is getting a lot of attention. Because GeoXACML is extended from XACML, we can apply results from related studies on XACML to GeoXACML. We discuss a few of those works here. In [3], Hounder solves the anomaly detection problem by transforming the target of rules into n-dimensional rectangles. After that, these super-rectangles are checked intersection for each dimension. If there are two super-rectangles intersect each other for all dimensions, we conclude that the corresponding rules of those two super-rectangles are conflicting or redundant. Hongxin Hu et al. [4] proposed a policy-based segmentation technique using Binary Decision Diagram (BDD) to represent XACML policies; and their algorithms will perform set operations such as unions (∪), intersections (∩) and set differences (\) on BDD trees to detect anomalies. In [5], Agrawal et al introduced a method for anomaly detection for a general access control policy language which can be applied to XACML. Other research which focus on integrating XACML policy [6, 7, 8], and optimizing XACML [9, 10] are also orthogonal to our work. Unfortunately, none of these consider spatial data types and anomaly detection of spatial constraints.

In this paper, we introduce an algorithm based on the path-consistency algorithm to automatically check if a new rule may be conflicting or redundant to existing rules in the system. This helps policy designers focusing on abnormal rules to choose a correct combining algorithm or change the rules properly. The remainder of the paper is organized as follows. Section 2, we overview GeoXACML and RCC8, and briefly discuss how to use RCC8 for topological reasoning. Then, our algorithm is discussed in detail in Section 3. In Section 4, we give an example of how our algorithm works on particular GeoXACML policies. We also consider computational complexity of the algorithm in this section. In the next Section, we give a brief introduction about our tool to detect anomalies for GeoXACML policies and some experimental results of this tool. Finally, the last section represents the conclusion and our further work.

2. Preliminaries

2.1 GeoXACML overview

GeoXACML is a language designed to describe access control policies to spatial data. GeoXACML uses eight OGC spatial relationship functions which have been defined in [11] for the DE-9IM to specify the spatial conditions in XACML policies. These functions take two Geometric shapes and return Boolean value. They are contains, crosses, disjoint, equals, intersects, overlaps, touches, within as shown in figure 2.

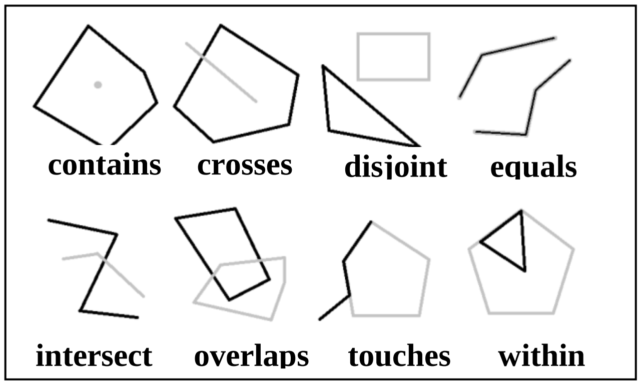

Figure 2 Eight OGC spatial relations used in GeoXACML.

2.2 RCC8 and Topological Inference

RCC8 is a relation algebra serving for representing and reasoning about binary relations between spatial regions in the RCC theory [12]. In this theory, regions are non-empty regular, closed subsets of a topological space, and multi-piece. RCC-8 has eight jointly exhaustive and pairwise disjoint basic relations denoted as: DC(disconnected), EC(external connected), PO(partially overlap), EQ(equal), TPP(tangential proper part), NTPP(non-tangential proper part) and their converse relations TPPi, NTPPi. All other relations are the union of two or more basic relations, or the empty relation. Hence, there are 256 different RCC-8 relations in all. A similar spatial relation model is the 9-Intersection

Method (9IM) of Egenhofer [13]. In this model, the author also defines eight relations with different names: disjoint, meet, equals, overlaps, covers, coveredBy, contains, inside. Table 1 shows the correlation between the OGC spatial relations and the relations in RCC8 and 9IM models.

A basic question in RCC8 reasoning is that what may be the relation of two regions x, z if we know the relation between x and y, and the relation between y and z. One technique for answering this question type is the composition table which is similar to the transition table used in Allen's work [14] on temporal reasoning. This table expresses the composition relation of two basic RCC8 rations R and S, written R ∘ S. The composition relation is the RCC8 relation which contains all RCC8 basic relation T such that R(x, y) ∧ S(y, z) ∧ T(x, z) is consistent, i.e. there is a spatial model. All composition results for basic relations are summarized in the RCC8 composition table (figure 4). This table first appeared in [15] in the context of GIS. An RCC8 relation can be expressed as a set of RCC8 basic relations {B1, B2,…Bn}. E.g. {EQ, DC, EC} means union of three basic relations: EQ, DC and EC. Then, the composition and converse of general relation is defined as follows:

$$R \circ R1 := \bigcup_{B \in R, B1 \in R1} B \circ B1$$

and $R^{\breve{}}$:={$B^{\breve{}}$:B∈R} (B, B1 are RCC8 basic relations; B∘B1 is shown in figure 4; $B^{\breve{}}$ is shown in figure 3). Other operations as intersection (∩), difference (\) are defined in the standard way [12].

OGC spatial relation	**RCC8**	**9IM**
equals	EQ	equals
disjoint	DC	disjoint
intersects	¬ DC	¬ disjoint
touches	EC	meet
within	NTPP,TPP	inside ∨ coveredBy
contains	NTPPi,TPPi	contains ∨ covers
overlaps	PO	overlaps
crosses	-	-

B	**B˘**
DC	DC
EC	EC
PO	PO
TPP	TPPi
NTPP	NTPPi
TPPi	TPP
NTPPi	NTPP
EQ	EQ

Figure 3 Equivalent Table (left) and Converse Relation Table (right).

2.3 Consistency and Path-consistency

The fundamental reasoning problem (called RSAT) in RCC8 is deciding consistency of spatial relations network Θ = {Rij(xi,xj) : 1 ≤ i, j ≤ n}, i.e. whether there is a model of Θ where the relations between the regions can be described by Θ. All reasoning problem can be reduced to it in polynomial time [16]. Unfortunately, RSAT is NP-complete [17, 18], i.e., no polynomial algorithm for deciding consistency.

B◦B1	**DC**	**EC**	**PO**	**TPP**	**NTPP**	**TPPi**	**NTPPi**	**EQ**
DC	DC, EC, PO, TPP, NTPP, TPPi, NTPPi, EQ	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC	DC	DC
EC	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPP, TPPi, EQ	DC, EC, PO, TPP, NTPP	EC, PO, TPP, NTPP	PO, TPP, NTPP	DC, EC	DC	EC
PO	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPP, NTPP, TPPi, NTPPi, EQ	PO, TPP, NTPP	PO, TPP, NTPP	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPPi, NTPPi	PO
TPP	DC	DC, EC	DC, EC, PO, TPP, NTPP	TPP, NTPP	NTPP	DC, EC, PO, TPP, TPPi, EQ	DC, EC, PO, TPPi, NTPPi	TPP
NTPP	DC	DC	DC, EC, PO, TPP, NTPP	NTPP	NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, EQ, TPP, NTPP, TPPi, NTPPi	NTPP
TPPi	DC, EC, PO, TPPi, NTPPi	EC, PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPP, TPPi, EQ	PO, TPP, NTPP	TPPi, NTPPi	NTPPi	TPPi
NTPP i	DC, EC, PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPP, NTPP, TPPi, NTPPi, EQ	NTPPi	NTPPi	NTPP i
EQ	DC	EC	PO	TPP	NTPP	TPPi	NTPPi	EQ

Figure 4 RCC8 Composition Table.

However, we can use the path-consistency algorithm to approximate consistency and to realize forward-checking [19]. This algorithm checks the consistency of all triples of relations and eliminates relations that are impossible though iteravely performing the operation Rij ← Rij ∩ Rik ◦ Rkj for all triples of regions i, j, k until a fixed point $\overline{R}$ is reached. If Rij = ∅ for a pair (i, j) then R is inconsistent, otherwise R is path-consistent. Path-consistent can be checked in $O(n^3)$ time [20]. This is achieved using a queue Q of couple of regions for the relations should be recomputed. Path-consistency does not imply consistency. The detail of this algorithm is shown in figure 5.

3. Anomaly Detection Algorithm

Each rule in GeoXACML can be transformed into a Boolean expression [21] and an effect. Each Boolean expression of a rule contains one or more atomic Boolean expressions combined by logical operators ∨ and ∧. Each atomic Boolean expressions is a logical function which returns either True or False (e.g. Subject="Doctor" or Time≤17:00 or within(Location, Polygon(0 0,10 0,10 10,0 10))). Two rules may be a conflict or a redundancy when the Boolean expressions of two rules can be true simultaneously. Therefore, to detect anomaly, we need to determine whether a conjunction of two Boolean expressions is satisfiable.

```
PATH_CONSISTENCY(C)
Input: A constraint network C
Output: A refined constraint network C*, True or False
01: $Q \leftarrow \{(i, j) \mid 1 \le i < j \le n\}$ // Initialize the queue
02: while Q is not empty do
03: select and delete an (i, j) from Q
04: for $k \leftarrow 1$ to n, $k \neq i$ and $k \neq j$ do
05:         $R \leftarrow Rik \cap (Rij \circ Rjk)$
06:         if $R \neq Rik$ then
07:                 if $R = \emptyset$ then
08:                         return False
09:                 $Rik \leftarrow R$
10:                 $Rki \leftarrow R^{\smile}$
11:                 $Q \leftarrow Q \cup \{(i, k)\}$
12:         $R \leftarrow Rkj \cap (Rki \circ Cij)$
13:         if $t \neq Rkj$ then
14:                 if $R = \emptyset$ then
15:                         return False
16:                 $Rkj \leftarrow R$
17:                 $Rjk \leftarrow R^{\smile}$
18:                 $Q \leftarrow Q \cup \{(k, j)\}$
19:return True
```

Figure 5 Path-Consistency Algorithm.

There are many challenges in determining whether a Boolean expression is satisfiable. This was the first known example of an NP-complete problem [22]. That briefly means that there is no algorithm that efficiently solves all instances of this problem, and it is generally believed (but not proven) that no such algorithm can exist. However, access control policies from many different application domains such as network, storage, database, and identity management often involve only the subclasses of Boolean expressions for which the satisfiability problem is tractable. In [5], the authors identify categories of Boolean expressions that occur frequently in typical policies and address the satisfiability problem for such cases. These categories consider functions {=, <, ≤, >, ≥} for real, integer, string, or calendar, Boolean data type and regular expression constraints.

In this paper, we focus on spatial constraints which uses 7 OGC spatial relations above (except crosses because it isn't supported in RCC8). We will only consider Boolean expressions that are conjunctions of clauses, where each clause is a disjunction of one or more atomic statements, all involving the same pair of objects (e.g. touches(x, y) ∨ overlaps(x, y) ∨ within(x, y)). Furthermore, without loss of generality, there is exactly one clause involving each pair of objects. If there are two or more clauses, it is equivalent to a disjunction of common basic relations of all clauses. If there is none, we consider it as the disjunction of all basic relations. We call such expressions topological expressions. The main problem here is to determine whether a topological expression is satisfiable.

Our anomaly detection algorithm is used when a rule is added or updated. The main idea is to iterate all existing rules in the system. For each rule, we call TOPOLOGICAL_CHECK_RULE function to check if the new rule is conflicting or redundant to the current rule. TOPOLOGICAL_CHECK_RULE function gets two rules as input and returns True if two rules may be conflicting or redundant, False otherwise. This function consists of following steps:

Step 1: build the Boolean expression

The Boolean expression is the conjunction of the Boolean expressions of these two rules. Here we just consider spatial relation constraints; other constraints are omitted.

Step 2: get real relations of all constant region couples.

Notice that regions in GeoXACML constraints can be constants. For each two constant regions, we can find out the real relation between them and then add those relations to the Boolean expression in Step 1 by conjunctions.

Step 3: convert the Boolean expression to RCC8 language.

We normalize the conjunction expression in Step 2 to topological expression as defined above. There is exactly one clause involving each pair of objects. If there are two or more, then this is equivalent to the disjunction of the relations that are common to all clauses. If the common relations are empty we will stop the algorithm and conclude that this RCC8 expression is unsatisfiable and two rules are not conflicting or redundant. For each couple of objects, we convert their relation from OGC relations into RCC8 by using the equivalent representation table in Table I.

Step 4: build the constraint network R

We build the constraint network matrix of the RCC8 expression in Step 3. Each cell Rij of this matrix contains the RCC8 relation of two object i, j in the RCC8 expression. Of course, Rji = Rij˘, Rii(diagonal of the matrix) is not used in this algorithm.

Step 5: check path-consistency of the constraint network R

PATH_CONSISTENCY function is called to approximate consistency of constraint network R.

Step 6: report the result.

The result of step 5 is either True, or False. If result is true, it means that two rules are potentially conflicting or redundant in spatial aspect. If result is false, two rules are surely separate, and there is not any request can be applied by both two rules.

4. Example And Analysis

4.1 Example

To clarify how our algorithm works, we will consider a simple example with two rules.
Rule R1:

(Subject=“Doctor” ∨ Subject=“ Surgeon”) ∧ (Action=“Read”) ∧ (Resource=“PatientRecord”) ∧ (8:00≤Time ∧ Time≤17:00) ∧ within(Location, Polygon(0 0,10 0,10 10,0 10)) → Permit

Rule R2:

(Subject = “Doctor” ∨ Subject = “Nurse”) ∧ (Action=“Read”) ∧ (Resource=“PatientRecord”) ∧ within(Location, Polygon(0 10,10 10,10 20,0 20)) → Deny

In this example, we need to check if rule R1 and rule R2 are conflicting or redundant or not. Then, we need to check if the following Boolean expression is satisfiable:

(Subject=“Doctor” ∨ Subject=”Surgeon”) ∧ (Action=“Read”) ∧ (Resource=“PatientRecord”) ∧ (8:00≤Time ∧ Time≤17:00) ∧ within(Location, Polygon(0 0,10 0,10 10,0 10))∧ (Subject = “Doctor” ∨ Subject=“Surgeon”) ∧ (Action=“Read”)∧ (Resource=“PatientRecord”) ∧ within(Location, Polygon(0 10,10 10,10 20,0 20))

Step 1: get the Boolean expressions. Here we just consider spatial constraints

within(Location, Polygon(0 0,10 0,10 10,0 10)) ∧ within(Location, Polygon(0 10,10 10,10 20,0 20))

Step 2: We have 3 objects in the topological expression:
x = Location, y = Polygon(0 0,10 0,10 10,0 10), z = Polygon(0 0,10 0,10 20,0 20). With 2 objects y, z are constants, we determine that their real relation is touches. We have new Boolean expression:

within(x, y) ∧ within(x, z) ∧ touches (y, z)

Step 3: Now we convert the expression in step 2 to RCC8 language as below

{NTPP, TPP}(x, y) ∧ {NTPP, TPP} (x, z) ∧ {EC}(y, z)

Step 4: build the constraint network for this RCC8 expression as below

	x	y	Z
x	-	NTPP,TPP	NTPP,TPP
Y	NTPPi,TPPi	-	EC
Z	NTPPi,TPPi	EC	-

Step 5: call PATH_CONSISTENCY with Constraint Network R as input:
First, we add all couple of objects to queue Q

$$Q \leftarrow \{xy, xz, yz\}$$

Get the couple xy from Q;

$$Q \leftarrow \{xz, yz\}$$

For object z:

$$Rxy \circ Ryz = \{NTPP, TPP\} \circ \{EC\} = \{NTPP\} \circ \{EC\} \cup \{TPP\} \circ \{EC\} = \{DC\} \cup \{DC, EC\} = \{DC, EC\}$$

$$R = Rxz \cap Rxy \circ Ryz = \{NTPP, TPP\} \cap \{DC, EC\} = \emptyset$$

With R = $\emptyset$, PATH_CONSISTENCY will return False

Step 6: the networks constraint R does not satisfy path-consistency. Therefore, we conclude that two rule R1, R2 cannot have conflicting or redundant relationship.

4.2 Correctness and performance analysis

Because path-consistency does not imply consistency, but consistency implies path consistency, TOPOLOGICAL_CHECK_RULE can return false positive, but never false negative. It means that if CHECK_RULE return False, then the policy designer can be sure that two checked rules cannot be conflicting or redundant. But, if TOPOLOGICAL_CHECK_RULE return True, this can be a false alarm.

This algorithm iterates all rules in the system and call TOPOLOGICAL_CHECK_RULE for each. TOPOLOGICAL_CHECK_RULE consists of some steps, but the most complex step is to check path-consistency in time $O(n^3)$. Therefore, the overall runtime of our algorithm is $O(n^3m)$; n is the average number of spatial objects in each rule's Boolean expression; and m is the number of rules in the system.

As analyzed in [18, 20], RSAT problem is NP-complete, it means that there is no algorithm which solve very large instances of this problem in reasonable time, but it is still possible to solve this problem with certain size of the input in reasonable time. Therefore, in step 5 of TOPOLOGICAL_CHECK_RULE function, we can use the backtracking search algorithm [18, 20] to check constraint network consistency instead of PATH_CONSISTENCY. This increases the computational time, but reduces the false alarm rate.

5. Implementation And Evaluation

5.1 Implementation

We implemented a tool to detect anomalies for GeoXACML. This tool is called by Policy Administration Point (PAP) when administrators add or edit a policy. It can detect anomalies for all types of data such as real number, integer, temporal types as well as topological types. It is based on the Conflict Detection Algorithm which is proposed by Hounder in [3]. The main idea of this algorithm is to transform policies as n-dimensional rectangles in n-dimensional space. Every dimension in the n-dimensional space is formed by an attribute in a policy. Two policies are abnormal if the intersection of two super-rectangles corresponding to them is not empty. Two rectangles intersect each other if every dimension intersects each other for all. To determine if two rectangles intersect each other for a dimension, we choose one of the following algorithms depending on data types and functions applied to the attribute corresponding to that dimension. We separate them into three expression types for each attribute:

+ Range restriction expression: this expression type contains attributes which are real number, integer, boolean, temporal data types, as well as string and applied comparison functions (equal, less, greater). AABB - AABB Intersection Algorithm is used to determine intersection for this expression type. The details of this algorithm are described in [3].

+ Regular expression: this expression type contains attributes which types are string (string, anyURI, x500Name, rfc822Name, ipAddress, dnsName) and there exists at least one *-match function among all policies that contain the same attribute. The Regex Intersection Algorithm is used to determine intersection for this expression type. The details of this algorithm are described in [3].

+ Topological expression: as shown above, this expression type contains spatial data types and functions. The main algorithm shown above is used to determine intersection for this type.

Figure 6 shows the main components of the Anomaly Detection tool: Anomaly Detector, Intersection Checker, RCC8 Reasoner. Anomaly Detector module gets a policy as the input; after that it loads all policies in the policy repository and checks if new policy has anomalies with existing policies. For each pair of policies, this component determines the intersection of two expressions corresponding to each attribute in two polices in turn by using sub-component of Intersection Checker. For case of topological expression, TOPOLOGICAL_CHECK_RULE calls RCC8 Reasoner component to determine if two topological expressions are intersect or not. After checking the intersection for each attribute, Anomaly Detector will combine results to obtain the list of policies which may be conflicting or redundant to new policy.

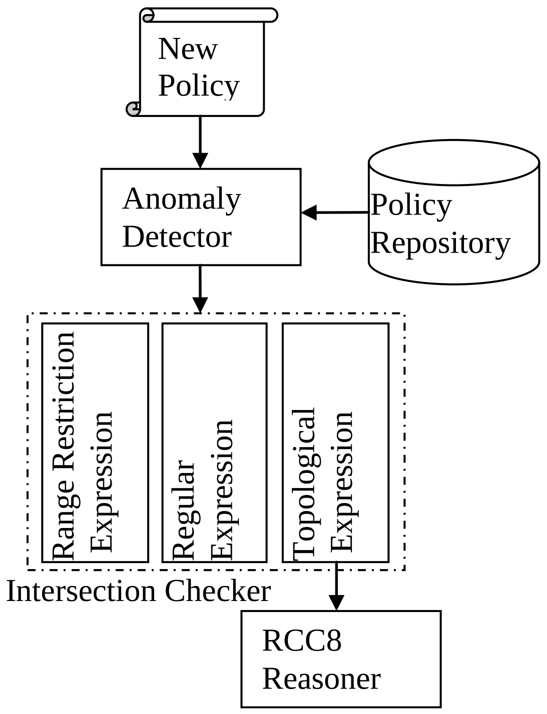

Figure 6 Anomaly Detection Tool’s Components.

5.2 Experimental Evaluation

To evaluate the effectiveness of the proposed algorithm, the experimental evaluation was conducted. As analyzed above, the main algorithm in this paper mainly depends on the number of policies and the number of spatial objects of each policy. We will compare the execution time between using Path-consistency and Consistency when changing the number of policies or the number of spatial objects of each policy. This test is conducted on a computer with CPU Intel Core i5 2.5GHz, 8GB RAM and Windows 7, 64bit OS.

Experiment 1: This experiment compares the execution time between applying Path-consistency and Consistency in Step 5 of the main algorithm when increasing the number of policy. In this experiment, the system contains an increasing amount of policy from 100 to 1000. Each policy has the form of R1 (RegionA, Polygon ((0 0, 10 0, 10 10, 0 10, 0 0))). The new policy has the form of R2 (RegionA, Polygon ((0 10, 10 10, 10 20, 0 20, 0 10))). In which, relations R1, R2 are randomly generated from the set of seven OGC relations: equals, disjoint, intersects, touches, within, contains, overlaps, and the negation of their relationship. Figure 7 shows the relationship between the execution time and the number of policies. It notices that the execution time of using both Path Consistency and Consistency algorithm increases slowly when the number of policies increases. The execution time of using Consistency algorithm is longer and increases more rapidly. However, the execution time of using Consistency is still acceptable for the case of small number of policies as in this experiment.

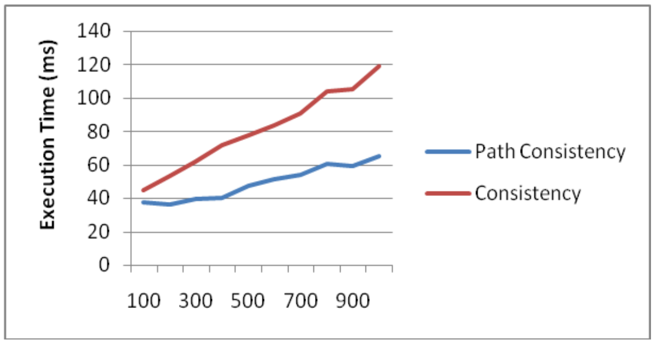

Figure 7 Execution Time for different number of policies.

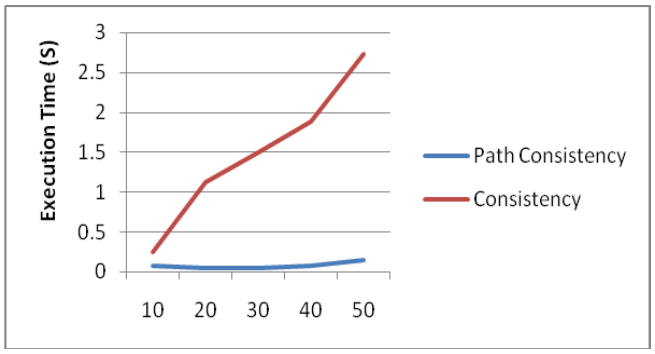

Figure 8 Execution time for different amount of spatial.

Experiment 2: we will compare the execution time between applying Path Consistency algorithm and Consistency algorithm in Step 5 of the main algorithm when the number of spatial objects in each policy increases. In this experiment, the system contains only one policy which contains an increasing number of spatial objects from 10 to 50. To show clearly the difference between using Path-consistency and Consistency, we need to generate constraints so that the final expression satisfies Path-consistency, then Consistency algorithm needs to check more property than Path-consistency. Therefore, in this experiment, the relations of spatial objects are either intersects or empty. Figure 8 shows the relationship between the execution time and the number of spatial objects. It is found that the execution time of using Consistency algorithm increases much faster than using Path-consistency algorithm. So when the number of objects increases, the execution time of using the Consistency algorithm is not acceptable.

From these experiments, it is recognized that the execution time of applying Path Consistency and Consistency algorithms has a big difference when the number of spatial objects increases. However, in case of simple constraints, with small number of objects, we can use Consistency in step 5 of the algorithm TOPOLOGICAL_CHECK_RULE to reduce the false alarm rate.

6. Conclusion And Future Work

In this paper, we proposed an algorithm based on the path-consistency algorithm to detect couples of rules which may be conflicting or redundant. This algorithm will be called when the policy designer add a new rule to the system or update an existing rule. These anomalies, after that, would be resolved by human administrators either by changing rules or choosing a properly combining algorithm for them. Our algorithm is based on the results of topological reasoning theory to detect anomalies in spatial constraints in GeoXACML policies. The algorithm is polynomial time. We also illustrated how the algorithm works through a simple example.

As mentioned above, although deciding consistency is NP-complete, but it is possible to solve its instances up to a certain size in reasonable time. Therefore, in the future, we will consider algorithms solving consistency problem to improve our algorithm for better false-rate. Moreover, in this work, we consider spatial constraints only; there is still further work to combine this algorithm with other algorithms which solve satisfiability problem for other Boolean expression types.

References

1. Matheus, A., Geospatial extensible access control markup language (GeoXACML) version 3.0. OGC implementation standard, Open Geospatial Consortium (OGC), March 2007.
2. Moses, T., eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS implementation standard, OASIS, February 2005.
3. Hounder, F., Conflict detection and resolution of XACML policies. Master's thesis, University of Applied Sciences Rapperswil, July 2010.
4. Hu, H., Ahn, G., and Kulkarni, K., Anomaly discovery and resolution in web access control policies. in Proceedings of the 16th ACM symposium on Access control models and technologies - SACMAT'11 ACM, 2011, 165–174.
5. Agrawal, D., Giles, J., Lee, K., and Lobo, J., Policy ratification. In Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, 2005, 223–232.

6. Mazzoleni, P., Crispo, B., Sivasubramanian, S., and Bertino, E., XACML Policy Integration Algorithms. ACM Transactions on Information and System Security, 11(1), 2008.
7. Ni, Q., Bertino, E., and Lobo, J., D-algebra for composing access control policy decisions. in Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ACM, 2009, 298–309.
8. Rao, P., Lin, D., Bertino, E., Li, N., and Lobo, J., An algebra for fine-grained integration of xacml policies. in Proceedings of the 14th ACM symposium on Access control models and technologies, ACM, 2009, 63–72.
9. Liu, A., Chen, F., Hwang, J., and Xie, T., Xengine: A fast and scalable xacml policy evaluation engine. ACMSIGMETRICS Performance Evaluation Review, 36(1), 2008, 265–276.
10. Marouf, S., Shehab, M., Squicciarini, A., and Sundareswaran, S., Statistics & Clustering Based Framework for Efficient XACML Policy Evaluation. in IEEE International Symposium on Policies for Distributed Systems and Networks, IEEE, 2009, 118–125.
11. Clementini, E. and Di, F. P., A Comparison of Methods for Representing Topological Relationships. Information Sciences - Applications, 3(3), 1995, 149-178.
12. Randell, D. A., Cui, Z., and Cohn, A. G., A Spatial logic based on regions and connection. in Proceedings of the 3rd. International Conference on Knowledge Representation and Reasoning, Morgan Kaufmann, San Mateo, 1992, 165-176.
13. Egenhofer, M. J. and Herring, J. R., A Mathematical Framework for the Definition of Topological Relationships. in Proceedings of the 4th International Symposium on Spatial Data Handling (SDH 90), Zurich, 1990, 803-813.
14. Allen, J.F., Maintaining knowledge about temporal intervals. Communications of the ACM, vol.26, 1983, 832–843.
15. Egenhofer, M. J., Reasoning about Binary Topological Relations. in Proceedings of the Second Symposium on Large Spatial Databases (SSDÕ91), Zurich, 1991. Lecture Notes in Computer Science no.525, 143-159.
16. Golumbic, M.C., and Shamir, R., Complexity and algorithms for reasoning about time: a graph-theoretic approach. Journal of the ACM, 40(5), 1993, 1108– 1133.
17. Grigni, M., Papadias, D., and Papadimitriou, C., Topological inference. in Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, 901–906.
18. Renz, J. and Nebel, B., On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. Artificial Intelligence,1999,69–123.
19. Mackworth, A., and Freuder, E., The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. Artificial intelligence, 25(1), 1985.
20. Renz, J., Nebel, B., Efficient methods for qualitative spatial reasoning. Journal of Artificial Intelligence Research, 15(1), 2001, 289-318.
21. Anderson, A., Evaluating XACML as a policy language. Technical report, OASIS, Mar 2003.
22. Cook, S.A., The complexity of theorem proving procedures. in Proceedings of 3rd Annual ACM Symposium on Theory of Computing, ACM, New York, 1971, 151-158.