Proc. IEEE Military Communications Conference, MILCOM2014, Baltimore. MD, Oct. 6-8, 2014

Dynamic Policy Enforcement using Restriction Set Theoretic Expressions (RSTE)

S. Yousaf Shah and Boleslaw K. Szymanski
Department of Computer Science and Network Science and Technology Center (NeST)
Rensselaer Polytechnic Institute (RPI), Troy, NY, USA
Email: {shahs9, szymab}@rpi.edu

***Abstract*—Service Oriented Sensor Networks consist of various assets and host variety of services, some of which are composed of other services. Policies are widely used for regulating access to assets and services specially when these assets are owned by different parties in a coalition environment. In this paper, we present a novel mechanism for policy implementation to provide or restrict access to resources using policies. We present "Restriction Set Theoretic Expressions (RSTE)" to represent assets and policies in the form of sets at system level, therefore *RSTE* is independent of high-level representation of policies and assets. High-level representation of network assets and policies can be easily translated to semantically defined *RSTE* sets and then different *RSTE* operations are applied to restrict or release access to resources. *RSTE* defines sets and operations that can be performed on the sets to implement policies. We describe semantics of *RSTE* sets and operations for assets in service configuration in WSNs and show how the services and policies can be represented as sets. We then leverage the capabilities of relational databases by representing sets as tables and applying policies as set operations executed as SQL queries. Operations performed on the database tables yield restricted sets of policy enforced services. Such services can then be provided to the user or used by service configuration to compose complex services. If service composition cannot be performed due to policy restrictions, the restricting conditions are reported to user through presentation layer for policy negotiation and relaxation.**

***Keywords*- Sensor Networks; Policies; Service-Oriented Architecture; Service Composition; Service Configuration;**

I. Introduction

Various languages and frameworks have been proposed for policy description, management and application [1], [2]. Every approach has its features and shortcomings. Recently, research on expressing and managing policies in human understandable language is getting significant traction. Researchers have used controlled natural language (CNL) to express user requirements and efforts are being made to devise frameworks that can manage policies using human readable languages [3]. However, translating these requirements from human readable format into system level constraints is a challenging issue. In general, policy representation frameworks tightly couple the actual implementation of policies with the higher level policy representation. This tight coupling leaves minimal flexibility for using alternative low-level policy implementation. Policy languages that are highly humanly expressive such as CNL based policies, have lower policy enforcement capabilities. On the other hand policy languages that support very sophisticated operations on system level are coupled with representation layer which is very programming oriented and too complex for an end user. Therefore, there is a need for decoupling the presentation layer from low level policy execution, so that policies can be expressed in user friendly format at presentation level and can be easily transformed into lower level policies for actual enforcement. Thus, maintaining a required level of semantic compatibility, the low-level policy enforcement and higher level representation should be as independent as possible to achieve design goal at two separate levels.

In this paper, we consider system for service configuration in a mobile sensor network at the application layer[4]. Each service hosted in the sensor network produces one or more outputs and requires zero or more inputs from other services, thus composite services are hierarchical in nature forming a workflow graph for data processing. In a coalition environment, services are owned by different collaborating organizations or partners, these services are hosted on sensor nodes owned by different partners. Policy makers define policy rules to restrict or allow access to the resources in the network. Users make requests for services with certain requirements, the requests are handled by the configuration system and based on policies and users' authorization corresponding services are configured for the requestor. In this paper, we show how *Restriction Set Theoretic Expressions (RSTE)* works for such a service configuration scenario. Due to space limitation details of *Application Layer* and controlled english based *Presentation Layer* are not provided.

As we focus on producing policy complaint services for mission specific applications [4]. Such missions include, multi-organization led rescue and recovery missions, military mission, monitoring and surveillance etc. Policy enforcement in application used in such dynamic scenarios are more challenging than traditional policy enforcement where all assets are indigenously owned by a single organization and the flow of data is pre-defined for applications. In dynamic service

Research was sponsored by US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

composition, the service graph is not pre-known as the status of the mission at a particular time defines the architecture of the services, moreover, services which are owned by different organizations dynamically appear and disappear during the course of a mission. Another important aspect of such systems is the nature of the policies, policies in such cases can be temporal in nature and as they are updated/negotiated by policy makers the effect should be instant. So, the challenges are: (1) how to policy enforce such dynamic services with minimal delay after a change in policy and report over restricting conditions for negotiation, (2) how to make policy description and management user friendly without overly limiting the capabilities of policy enforcement engine. The main contributions of this paper are,

1) A mid-layer RSTE language with semantics and operations for policy enforcement.
2) Utilization of relational algebra and relation database for policy enforcement using *RSTE*.

The Remainder of the paper is organized as follows, in Section II, we present background and an application scenario for *RSTE*. In Section III, we provide detailed description of *RSTE* and Section IV provides implementation details on how *RSTE* can be implemented using relational database. Finally, Section V concludes the the paper.

II. Application Scenario and Related Work

A. Emergency Response Coalition Scenario

Imagine an emergency response coalition formed between US and UK forces to evacuate a disaster stricken area. A sensor network is setup to monitor the area as well as to provide support to the rescue forces. Network assets are owned by either US or UK, therefore, sharing policies have been defined to allow partners to access each others' resources. As other nations join the rescue mission, more policies are defined to manage resources in coalition. Resources are not always equally accessible to all partners but as the situation evolves policies are defined to provide access to coalition partners. We show policy enforced services sharing using *RSTE* for such a scenario.

B. Related Work

Asset sharing and management in coalition environments has been studied in the past and various techniques for resource allocation have be suggested by researchers [5], [6]. Policy enforcement and management techniques for different computing paradigms have been proposed in the past [7], [8], [9]. In [8], [9], more programatic and standardized approaches have been proposed, however, the policy description and representation is closer to computer programs thus not very friendly to non-technical users. Goal oriented and formal logic based frameworks have also been propped in the past for policy management [10], but their capabilities are limited by the underlaying OWL version. Other semantic representation based frameworks such as [11], [12] have also been proposed. Deontic logic based approaches to address paradoxes in Standard Deontic Logic (SDL) have been proposed in [13], however, our approach differ from deontic logic based approaches and other formal logic based approaches such as [14], [10] in policy representation as well as enforcement because the latter are not feasible for dynamic compositions of services. In [2] attribute based policy enforcement has been investigated, but this approach also utilizes [8] and has more programming approach for policy representation, also it does not support backtracking and suggestions for policy negotiation.

III. The **RSTE** Language

We present a *Restriction Set Theoretic Expressions (RSTE)* language, that is simple yet expressive enough to map user requirements to lower level system constraints so that lower level execution engines can produce desired result. This set language is understandable by both higher layers of framework dealing directly with the user and low level system engines. Therefore, this language enables two way information flow from user to the system level objects and back. Figure 1 shows layered view of three major components of the system.

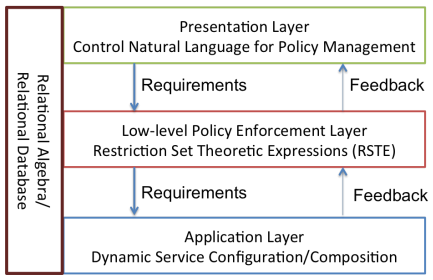
Relational Algebra/
Relational Database
Presentation Layer
Control Natural Language for Policy Management
Requirements
Feedback
Low-level Policy Enforcement Layer
Restriction Set Theoretic Expressions (RSTE)
Requirements
Feedback
Application Layer
Dynamic Service Configuration/Composition

Fig. 1: Set Language is incorporated as layer between Presentation layer and Application

For high level of user friendliness in our approach, we propose that policies are represented in Controlled English [3] as Controlled Natural Language (CNL) on presentation layer, as shown in Figure 1, however this is not a requirement for *RSTE*. Policies can be represented in any language at *Policy Presentation Layer*, but the policies and constraints should be compliant with or transformable to semantics of *RSTE*.

The *CNL for Policy Management* layer is responsible for policy management on presentation layer. Policies are defined and modified at this level. If there are any conflicts between different policies, this layer also resolves them by running conflict resolution algorithms. Moreover, this layer performs negotiations in case conflicts cannot be resolved between policies or no valid system configuration can be produced under specified policies.

The *Dynamic Service Configuration/Composition* layer is responsible for producing policy enforced low cost composite services based on user requirements. This layer deals with input/output matching of different services as well as spatial relevancy constraints to produce composite services that are cost effective and geo-spatially relevant to the user's interests. In case when this layer fails to produce service configuration requested by the user, it notifies the *Controlled*

Natural Language for Policy Management via *RSTE* layer about the need for appropriate actions. The *Dynamic Service Configuration/Composition* layer accesses resources that are policy constrained by the *RSTE* layer, therefore this layer considers only resources that are accessible by the requesting user in compliance with policies. If the resources are too scarce for service composition, it generates corresponding output message that is sent to the presentation layer.

A. The Semantics and Operations of ***RSTE*** Language

In this section we define *RSTE* operations and semantics for *RSTE* sets. A request for service is represented by $SReq$ set and its corresponding service response is represented by $SRes$. Following are the sets used by *RSTE* for representing assets, policies, requests/response and users. Alongside these sets we define various set operations, such as *Union, Intersection, Subtraction, Cardinality*, that are performed on these sets to produce a restricted set of assets which is then provided to the system layer for configuration.

$Service\ Set = SS = \{\{ServiceName, NodeId, GeoSpatialCoverage, Capabilities, Ownership\}\}$

$Node\ Set = NS = \{\{NodeId, Ownership, Location, Conditions, Permissions\}\}$

$Service\ Request\ Set = SReq = \{\{RequestId, UserId, ServiceName, Capabilities, Properties, Restrictions\}\}$

$Policy\ Set = PS = \{\{PolicyId, ServiceOwnership, ServiceName, Conditions, UserAffiliation, Restrictions, Action\}\}$

$User\ Set = US = \{\{UserId, UserName, UserAffiliation, Role, UserProperties\}\}$

$RSTE\ Response\ Set = RRS = \{\{ServiceName, NodeId, GeoSpatialCoverage, ServiceCapabilities, ServiceOwnership, PolicyId, PolicyConditions, UserAffiliation, PolicyRestrictions, RequestId, UserId, SReqCapabilities, SReqRestrictions, Action\}\}$

$Service\ Response\ Set = SRes = \{\{RequestId, Logs, ReturnValue, Failures\}\}$

B. Definitions of the Sets

1) *Service Set (SS)*: This set represents the snapshot of the available services. This set is provided by the upper layer i.e., the *CNL Layer*. This set is subset of all the services hosted in the asset database. During the set operations this set is filtered based on policies and restrictions applied by the user.
2) *Node Set*: This set represents nodes that host member services of *SS*. This set is used to access detailed information about the infrastructure hosting the service and will be used as needed.

TABLE I: Service Set

Set Element Name	Description
Service Name	Name of the service, e.g., LOBR
NodeId	Unique ID of node hosting the service
GeoSpatialCoverage	Geospatial coverage provided by the service
Capabilities	Capabilities provided by the service, e.g, {<Sensor, Video>, <Resolution, HD>}
Ownership	Organization that owns the service, e.g., US.

TABLE II: Node Set

Set Element Name	Description
NodeId	Unique ID of node hosting the service
Ownership	Organization that owns the node, e.g., UK.
Location	Physical Location of the Node, e.g., $< LAT, LONG, ALT >$
Conditions	Conditions that applies to usage of the Node.
Permissions	Permissions on this node

3) *Service Request Set*: This set describes service requests made by users.

TABLE III: Service Request Set

Set Element Name	Description
RequestId	Unique ID of the service requests to distinguish different requests.
UserId	Unique IDs of the users.
ServiceName	Name of the service that has been request.
Capabilities	Capabilities that are required by the service, e.g., High resolution video
Properties	Configuration settings of the service e.g., {<ModeofOperation, Distributed>}
Restrictions	Restrictions on the requested service, e.g., {<Ownership, US>}

4) *Policy Set*: The *Policy Set* represents policies that need to be applied to the *SS* using various *RSTE* operations. This set captures various other aspects of the policies (e.g., restrictions) along with *conditions* and *actions* policy enforcement paradigm.

TABLE IV: Policy Set

Set Element Name	Description
PolicyId	Unique ID of a policy
ServiceOwnership	Owner organization of the service affected by the policy, e.g, UK.
ServiceName	Name of the service affected by the policy.
Conditions	Conditions on the use of the service, e.g., {#service instances < 5}
UserAffiliation	Affiliation of the user requesting the service, e.g., US or UK
Restrictions	Restrictions on the use of the service, e.g., $\{< Role, Commander >\}$
Action	Access to service based on the policy, i.e., Allow or Deny

5) *User Set*: User Set describes user of the network. Each user has certain properties as well as associated orga-

nization. The *UserId* enables the set to access more detailed information about the user from the detailed assets database in case details are needed. This set is primarily used for policy validation through *RSTE* operations. Various conditions are matched based on the user profile before authorizing user to access services.

TABLE V: User Set

Set Element Name	Description
UserId	Unique ID of a user.
UserName	Name of the user.
UserAffiliation	Affiliated organization of the user, e.g., US
Role	Role of the user in organization or in a mission, e.g., $< Role, Commander >$
UserProperties	Other properties of the user, e.g., $< SkillSet, Expert >$

6) *RSTE Response Set*: The *RSTE Response Set* is the set that is supplied to system layer along with *Service Request Set (SReq)* for final service configuration. This set contains services that fulfill the policy and user requirements.

TABLE VI: RSTE Respnse Set

Set Element Name	Description
Service Name	Name of the service, e.g., LOBR
NodeId	Unique ID of node hosting the service
GeoSpatialCoverage	Geospatial coverage provided by the service
ServiceCapabilities	Capabilities provided by the service, e.g., {<Sensor, Video>, <Resolution, HD>}
ServiceOwnership	Organization that owns the service, e.g., UK.
PolicyId	ID of policy applied to this service.
PolicyConditions	Conditions from policy applied to this service.
UserAffiliation	Affiliation of the user, e.g., US, UK.
PolicyRestrictions	Restrictions specified by policy.
RequestId	ID of the user's request.
UserId	ID of the user of this service.
SReqCapabilities	Capabilities requested in the Service Request
SReqRestrictions	Restrictions specified in the Service Request.
Action	Authorization based on policy.

7) *Service Response Set*: The service response set is primarily for feedback purposes. This set is supplied to the *CNL Layer* in order to be presented to the user or to negotiate/relax policies.

TABLE VII: Service Response Set

Set Element Name	Description
RequestId	Unique ID of the service request
Logs	Any logs produced by the service configuration at the system level.
ReturnValue	Exit state of the service configuration, e.g, 0,1,-1.
Failures	In case of failures, reason of failure.

C. The RSTE Operations

The *RSTE* operations are like normal *Set Operations* but these operations can also be performed based on a specific condition (which can be imposed on set members), which is checked in the corresponding sets. Consider two sets A and B, members of these sets are also sets of same semantics. Let {e-1,e-2,e-3} be the semantics of sets A and B.
$A = \{\{1, 2, 3\}, \{a, b, c\}\}$
$B = \{\{1, 5, 10\}, \{e, f, g\}, \{a, x, y\}, \{a, b, c\}\}$

1) **Union** ($\cup$) The union operation combines two sets into one. An union operation is defined as,
$A \cup B = \{x : x \in A \; or \; x \in B\}$
A union with a condition ν can be written as,
$(A \cup B)_\nu = \{x \colon x \in A \; or \; x \in B \; and \; \nu = true\}$
A simple union of the two sets will be as follows,
$A \cup B = \{\{1, 2, 3\}, \{a, b, c\}, \{a, x, y\}, \{1, 5, 10\}, \{e, f, g\}\}$
A union operation defined on a condition "ν: (A.e-1=B.e-1)", produces following set,
$(A \cup B)_\nu = \{\{1, 2, 3\}, \{1, 5, 10\}, \{a, b, c\}, \{a, x, y\}\}$
2) **Intersection** ($\cap$) The intersection of two sets results into a set with common members among the sets.An intersection operation is defined as,
$A \cap B = \{x : x \in A \; and \; x \in B\}$
An intersection with a condition ν is written as,
$(A \cap B)_\nu = \{x \colon x \in A \; and \; x \in B \; and \; \nu = true\}$
A simple union of the two sets is,
$A \cap B = \{\{a, b, c\}\}$
An union operation defined on a condition "ν: (A.e-1=1)", produces an empty set.
$(A \cap B)_\nu = \{\}$
3) **Difference** ($-$) The difference or subtraction is the relative complement of set B in A. The difference operation is defined as,
$A - B = \{x : x \in A \; and \; x \notin B\}$
A difference operation with a condition ν is written as,
$(A - B)_\nu = \{x \colon x \in A \; and \; x \notin B \; and \; \nu = true\}$
A simple difference of the two sets will be as follows,
$B - A = \{\{1, 5, 10\}, \{e, f, g\}, \{a, x, y\}\}$
A difference operation defined on a condition "ν: (A.e-1=a)", produces set.
$(B - A)_\nu = \{\{a, x, y\}\}$
4) **Cardinality** ($|\;|$) Cardinality of a set gives the size of a set. In above example, $|A| = 2$ and $|B| = 4$.

D. Operations for Policy Restrictions

First we need to find out policies that apply to certain user, then we will be able to find out services that a specific user is authorized to access contingent to conditions and restrictions. Moreover, there may be services that are not policy enforced, such services should be available to all the users to use. In order to find policies that need to be applied to a service configuration request, we apply series of operations to the policy set (PS).

1) $PS_1 = (PS \cap US)_\nu \qquad \nu = UserAffiliation$
2) $PS_2 = (PS_1 \cap US)_\nu \; \nu = Role \wedge UserProperties$

3) $SS_{PolicyEnforced} = (SS \cap PS)_{\nu}$
$\nu = ServiceName \wedge ServiceOwnership$
4) $SS_{PolicyAllowed} = (PS_2 \cup SS_{PolicyEnforced})_{\nu}$
$\nu = ServiceName \wedge Ownership$

Now we find all those services, which have no restrictions on them or there is no policy defined to restrict them. These services are available to all the users. To find services with no policy restrictions, we perform following operations.

5) $SS_{NoPolicyEnforced} = (SS - SS_{PolicyEnforced})$
6) $SS_{AllAllowedServices} = (SS_{PolicyAllowed} \cup SS_{NoPolicyEnforced})$
7) $SS_1 = (SS_{AllAllowedServices} \cup SReq)_{\nu}$
$\nu = Capabilities$
8) $RSTE\_SRes = SS_1$ Process based on restrictions and conditions

IV. RSTE Implementation

We have implemented the prototype of *RSTE* language using a relational database. The *Relational Algebra* in Relational Database Management Systems (RDBMS) provides a natural support for *RSTE* sets and operations, therefore the capabilities of relational databases can be used for efficient implementation of *RSTE*. The sets in the language and tables in a relational database are analogous. We represent every set by a corresponding table. We then perform different operations on the tables using relational algebra and create views for intermediate steps. These views are then systematically processed to produce other views and finally the view that represents the *RSTE Response Set*, which is further processed before providing it to the service configuration layer. For some of the operations shown in section III-D, we have created "Views" in the database, whereas the operations that deal with temporal requirement or that need more fine grain processing e.g., Step-6 are done via post-processing on *RSTE Response Set*.

When a service request is received from the user, it is represented by *SReq* set and corresponding entry is made in the database table. The views automatically get updated and set of all allowed services are filtered out. Now, if the user has defined any specific restrictions or there are policies related to the user requesting a service, further processing is done via code, after which allowed services are provided to the *Application Layer* for service configuration. For certain services or users very restrictive sharing policies may be defined by the policy makers. Such overly restrictive policies limit the services available for configuration and these services might not be sufficient for a successful configuration leading to failure in meeting the user's request. In such cases, the *RSTE* uses backtracking mechanism to find out condition(s) that can be relaxed in order to produce a configuration. The condition(s) are then sent to policy management layer for policy negotiation. When such a condition is relaxed in the policy, the resources are immediately available to the user and service can then be configured enforcing the updated policy.

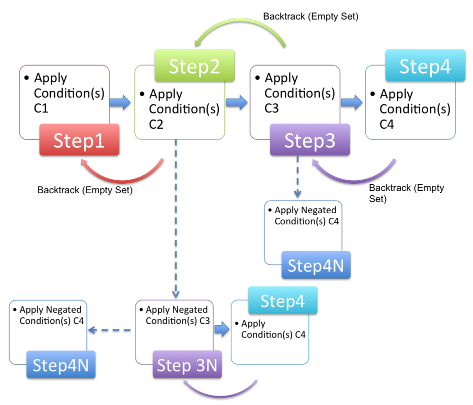

Fig. 2: Backtracking in for policy negotiation in case of overly restricted policies

A. RSTE Backtracking

Policies specified by policy makers might be very restrictive which may constrain resources necessary for service configuration. In a very restrictive case, operations performed in step by step policy application by RSTE may lead to empty sets or empty views in our implementation. Which means that there are no services available for configuration. At any such step where we detect an empty set we back track to previous step and negate the conditions to see if that leads to a non-empty set. If so we can suggest this negation to *CNL layer* as relaxation for polices involved. In case there are more than one conditions applied at a particular step, each condition is negated separately and the resultant set is checked for not being empty, this way relaxation can be performed at more granular level and only over-restrictive conditions will be negotiated/relaxed.

Figure 2 shows the backtracking mechanism in the *RSTE* implementation. As shown in the figure 2, when condition "C4" of the policy is applied at "Step4" all services are filtered out yielding an empty set; at this stage the policy enforcement backtracks and applies negotiation of condition "C4" to go to "Step4N". If at this stage the result set is not empty then condition "C4" is restricting the solution and it is propagated to *Presentation Layer* for policy negotiation.

B. RSTE Example

We evaluated RSTE implementation using a *RDBMS*. For initial evaluation, we populated various assets in the database including 24 services, 8 users, 6 nodes and 8 policies to test if *RSTE* returns policy enforced services. Here we present some simple rules in natural language for elaboration and show the response produced by *RSTE* after policy enforcement operations were performed on sets. The following example assumes that the transformation from *CNL* to *RSTE* sets is performed at the *Application Layer*.

Policy Rule: `If (user U isAffiliated with partner 'US') and (the user U has the value Role as UserRole) and (the value Role = 'Intel') and (service S hasName`

```
'CAMERASERVICE') and (partner 'US' owns
service S) then (the user U hasAccess to
service 'CAMERASERVICE')
```

User Request: A user "John Smith" requests for a service, the user is represented in *User Set* as follows,
$User\ Set = US = \{7, JohnSmith, US, Intel, Analyst\}$
User's request is represented by following set in the *RSTE*,
$SReq = \{4, 7, CAMERASERVICE, HDCAMERA, Distributed, None\}$
As soon as the above request is made by "John Smith", the system enforces policy and produces filtered services that this user is authorized to access based on user's particulars. *RSTE* produces following set as *RSTE_RESPONSE*.
$\{\{11, 11, Coverage = 5, HDCAMERA, US, 5, , US, Role = Intel, true, JohnSmith, 7, Role = Intel, Role = Analyst, 4, CAMERASERVICE, HDCAMERA, DISTRIBUTED, None\}\}$.
As shown above, there is only one service that is available for the user, this is because the policies in the system allowed only a subset of services to be available to this user and then the service request had restriction on the services to have capability of a "HDCAMERA" which further reduced the number of services to only one, which satisfies both policies and user's criteria.

To test backtracking mechanism, we modified the role of user "John Smith" from "Intel" to "Soldier". This produced an empty set and the user no longer can access "CAMERASERVICE" that he was able to use with "Intel" role. The backtracking mechanism backtracks to condition which leads to empty set. The backtracking is only performed on operations that filter services based on policies. So, when we ran queries with negated conditions, we found following query that restricts services based on roles. When that query was negated a non-empty set was produced which signified condition based on roles as over-restricting and the condition was reported for negotiation. The original query was

```
SELECT "PS"."ID", "PS"."PolicyId",
"PS"."ServiceOwnership", "PS"."ServiceName",
"PS"."conditions", "PS"."UserAffiliation",
"PS"."Restrictions", "PS"."Action",
"US"."UserName", "US"."UserId" FROM "PS",
"US" WHERE "PS"."UserAffiliation" =
"US"."UserAffiliation" AND "US"."UserName"
= 'John Smith' AND ( "PS"."Restrictions"
= "US"."Role" OR "PS"."Restrictions" =
"US"."UserProperties" );
```

when condition on roles is negated as

```
...WHERE "PS"."UserAffiliation" =
"US"."UserAffiliation" AND "US"."UserName"
= 'John Smith' AND not ( "PS"."Restrictions"
= "US"."Role" OR "PS"."Restrictions" =
"US"."UserProperties" );
```

the query produced results. The condition

```
"PS"."Restrictions" = "US"."Role" OR
"PS"."Restrictions" = "US"."UserProperties"
```

is reported for negotiation. This meets our expectations as we updated the role of John Smith from "Intel" to a "Soldier".

V. Conclusion

In this paper, we present a novel mechanism for dynamic policy enforcement on services in sensor networks. We present *RSTE* language and we show that the network assets, user requirements and policy constraints can be represented as sets. We then show that we can apply set operations to these sets to enforce policy constraints and report any overly strict constraints to the user. We demonstrate that *RSTE* can be implemented using *RDBMS* and various *Relational Algebra* operations can be utilized to implement *RSTE* operations. We show that *RSTE* can enforce policies and perform backtracking. In future, we aim to perform detailed evaluation of *RSTE* in combination with the controlled english as well as service configuration to test its capabilities for broad range of policies

References

[1] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," *Journal of Network and Systems Management*, vol. 15, no. 4, pp. 447–480, 2007.

[2] R. Dilmaghani, S. Geyik, K. Grueneberg, J. Lobo, S. Y. Shah, B. K. Szymanski, and P. Zerfos, "Policy-aware service composition in sensor networks," in *Services Computing (SCC), 2012 IEEE Ninth International Conference on*. IEEE, 2012, pp. 186–193.

[3] Ibm controlled english. [Online]. Available: https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=558d55b6-78b6-43e6-9c14-0792481e4532

[4] S. Y. Shah, B. Szymanski, P. Zerfos, C. Bisdikian, C. Gibson, and D. Harries, "Autonomous configuration of spatially aware sensor services in service oriented wsns," in *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom Demos)*. IEEE, 2013, pp. 312–314.

[5] T. Pham, G. H. Cirincione, D. Verma, and G. Pearson, "Intelligence, surveillance, and reconnaissance fusion for coalition operations," in *Information Fusion, 2008 11th International Conference on*. IEEE, 2008, pp. 1–8.

[6] A. Preece, T. Norman, G. de Mel, D. Pizzocaro, M. Sensoy, and T. Pham, "Agilely assigning sensing assets to mission tasks in a coalition context," *IEEE Intelligent Systems*, vol. 28, no. 1, pp. 57–63, 2013.

[7] G. Karjoth, "Access control with ibm tivoli access manager," *ACM Transactions on Information and System Security (TISSEC)*, vol. 6, no. 2, pp. 232–257, 2003.

[8] Ibm research, policy management library. [Online]. Available: https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=ed556565-1d91-4289-94ae-213df1340350

[9] J. Lobo, "Cim simplified policy language (cim-spl)," *Specification DSP0231 v1. 0.0 a, Distributed Management Task Force (DMTF)*, vol. 10, 2007.

[10] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, and A. L. Lafuente, "Using linear temporal model checking for goal-oriented policy refinement frameworks," in *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*. IEEE, 2005, pp. 181–190.

[11] M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara, "Owl-polar: A framework for semantic policy representation and reasoning," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 12, pp. 148–160, 2012.

[12] S. A. Chun, V. Atluri, and N. R. Adam, "Using semantics for policy-based web service composition," *Distributed and Parallel Databases*, vol. 18, no. 1, pp. 37–64, 2005.

[13] H. Prakken and M. Sergot, "Dyadic deontic logic and contrary-to-duty obligations," in *Defeasible deontic logic*. Springer, 1997, pp. 223–262.

[14] A. K. Bandara, E. C. Lupu, J. Moffett, and A. Russo, "A goal-based approach to policy refinement," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*. IEEE, 2004, pp. 229–239.