Knowedge-based Policy Conflict Analysis for Collabrative Workspace

Zhengping Wu
Department of Computer Science and Engineering, University of Bridgeport
221 University Avenue, Bridgeport, CT 06604, USA
zhengpiw@bridgeport.edu

Yuanyao Liu
Department of Computer Science and Engineering, University of Bridgeport
221 University Avenue, Bridgeport, CT 06604, USA
yuaoyaol@bridgeport.edu

***Abstract*— In collaborative workspace, collaborations are constrained by different requirements between different participants. Since policy-based management can be used to simplify collaboration management, administrators and users can use policies to define control rules and configurations of collaborative workspaces. These control rules and restrictions actually reflect management needs and business contracts. When collaboration is necessary between participants for a specific task, various management requirements from individuals may have conflicts. The situation is even worse when the collaboration is a one-time event. To detect and resolve potential conflicts within a collaborative workspace, a knowledge-based agent framework is proposed and used in this paper to analyze potential policy conflicts. Experiments in a sensor network environment confirm several advantages of the proposed framework.**

Keywords - policy conflict analysis, knowledge-based analysis, temporal logic, semantic extension, collaborative workspace.

I. INTRODUCTION

Collaborative workspace is a new trend for people to work together, which helps people overcome the geographic obstacles. There are also physical requirements for collaborative workspace, such as video and audio connections, and remote control sensors. Information in a collaborative workspace is usually shared through networks. In order to build an efficient and secure collaborative workspace, policy-based management can be used. Policy-based management is an administrative approach to manage system usage and its governance rules within an information domain. More and more systems have adopted this policy-based approach for its convenience and efficiency. Policies represent requirements in collaborative workspaces. A policy domain (domain hereafter) is a collection of elements and services administered in a coordinated fashion [1].

Collaborative workspace can support interactions and collaborations between participants. Different workspace participants can share their resources and build new workspace based on existing workspaces. For example, domain A has one participant and one resource. Participant "A" requires resource to provide certain functionality. Another domain B contains one participant and another resource. Participant "B" also needs to require some functionality provided by its resource. When these two domains "A" and "B" collaborate, they share their resources. At this point, the resource has two policies for participant "A" and "B" respectively. However, different management requirements of these services are reflected in different policies in a policy-based management environment. These requirements may conflict with each other, which are usually reflected in policy conflicts. For example, before domain A and domain B collaborate, there are policies to control their services and their own data services. We call the policy in domain A "Policy 1", the policy in domain B "Policy 2". In "Policy 1", sensor "D" has to provide data for participant "A", with an adaptive interval (the default value is 100s). This value is adjustable according to request from participant "A". In "Policy 2", resource "D" has to provide data under certain data error rate for "B". During the collaboration, Policy 1 and Policy 2 will have some overlap on sensor "D". If participant "A" requests high-speed data, this request may occupy a large amount of resources on sensor "D". Therefore, sensor "D" cannot provide qualified data to participant "B". When collaboration happens, there may be many conflicts between two policies, which may affect the whole collaboration. In this paper, we propose a knowledge-based policy conflict analysis framework to analyze policies and figure out such potential conflicts.

II. RELATED WORK

A. Temporal Logic

Properties of complexity and axiomatizations for temporal logics have been studied for decades. Temporal logics are widely used in the specification and verification of distributed systems. However, along with explosion of information, one element in different systems may carry number of attributes, and these attributes may assign this element different roles. Information becomes a barrier of specification and verification. In previous approaches [2, 3], information is presented as constraints. In [2], authors implemented DLTL in specifying and verifying systems of communicating agents and interaction protocols. The semantic facts of agent communication have been specified by means of laws and constraints. Authors in [3] provide a logical framework (Temporal Action Logic, TAL) for specifying and verifying systems of communicating agents and interaction protocols. This framework provides a simple formalization of the communicative actions in terms of their effects and preconditions, and the specification of an interaction protocol by means of temporal constraints. In [4], authors present a general framework and a specification language FCTL (first order CTL) for specifying properties in trust management systems. This framework focuses on

COLLABORATECOM 2012, October 14-17, Pittsburgh, United States

DOI 10.4108/icst.collaboratecom.2012.250514

dynamic policies, which may change their effectiveness at run-time. In [5], authors present a temporal logic with temporal constraint (Fuzzy Temporal Logic, FTL), which is used in supporting efficient query answering. In [6], authors present a generalization of temporal logics: CTL and the μ-calculus. Both extensions are defined over C-semirings, an algebraic structure that captures many problems and that has been proposed as a general framework for soft constraint satisfaction problems (CSP).

B. Conflict Analysis

Research in conflict analysis has been growing over years. Logic languages are widely used in this field. Temporal logic has been used to analyze properties in many types of policies. For example, First-order Temporal Policy-analysis Logic (FTPL) [7] is used to check whether a SPKI policy state satisfies a property specified in FTPL. This property check is for static properties and static policies, which is not sufficient for collaboration activities. A formal policy analysis framework to identify trusted computing base (TCB) with the consideration of specific security goals is proposed in [8]. Authors build an information domain for TCB policies, and use their rules to identify possible policy conflicts in a system. But this methodology cannot be applied in dynamic processes. Another logic-based policy analysis framework is proposed in [9], which uses Event Calculus to represent and reason about changing properties of a domain regulated by policies. Two interesting aspects of this framework are the runtime evaluation of policy rules and the offline analysis of policies accomplished by an abductive constraint approach. The runtime evaluation of policy rules is used to abate effective policy conflicts. Then offline analysis is used to detect policy conflicts. So this framework actually cannot resolve runtime conflicts if new policies are applied into the system at runtime. Neither can this framework detect dynamic conflicts. An empirical policy analysis tool implemented upon open source DL reasoner Pellet is described in [10]. Before Pellet analyzes policies, a mapping function will translate XACML rules into formalism. Then policy comparison, policy redundancy analysis and policy verification are performed by Pellet to identify conflicts. A general model of security policies has been discussed in [11]. Detection and reconciliation of security policy conflicts following that model are restrained by the complexity of the policy set. And only two-party conflict reconciliation can be tractable. Applications of the two-party conflict detection and reconciliation method to KeyNote [12] and GAA-API [13] systems are also discussed. But the capability for dynamic conflict detection and reconciliation is still missing.

III. Policy Analysis Using Temporal Logic

A. Temporal Logic

The primary feature of a logic theory is its order, which defines the domain of all formulae described by the logic. Propositional logic is based on a set of elementary facts by using a set of logic operators. It indicates a Boolean value set. First-order logic is an extension of propositional logic. Temporal logic assumes that facts hold at particular time or time points and they are ordered. Temporal first-order logic extends the first-order logic with a time dimension. It has been broadly used to cover temporal information within a logical framework.

Logical operators and expressions are used in different representation technologies. Usually, logical representations employ the notation of constant, variable, function, predicate, logical connective and quantifier to represent facts as logical formulae. In modal logic, the concepts of truth and falsity are not static and immutable, but are, on the contrary, relative and variable. In modal logic system, if V is the evaluation function for formulae, then it can be written as:

$$V : F \times S \rightarrow \{true, false\}$$

The F is the set of formulae, S is the set of states, and the V assigns a truth value to every formula in every state set. The V denotes whether there is a relationship between different states *s1* and *s2* that can be expressed by a formula *f*. Temporal logic is built as an extension of classic logic. Temporal logic adds a set of new operators likes ***H, G***:

***H*:** always in the past;

G: always in the future;

The formal definition is:

$$V(\mathbf{G}f, t) = true \quad iff \forall S \in T.t < s \Rightarrow V(f, s),$$

$$V(\mathbf{H}f, t) = true \; iff \forall S \in T.s < t \Rightarrow V(f, s),$$

A major drawback of logical representation is the lack of organizational principles for the facts constituting a knowledge base. Temporal logic also has this drawback. A knowledge base stores entities and relationships within a world into a database (this word "world" denotes a collection of objects. This object set includes entities and relationships among these entities.)

B. General Policy Model

In a policy-based management system, a policy describes several actions and information about these actions. In our previous works [14], we considered one policy define one action, and for each action, there is an executor or a type of executors, a target or a type of targets, and some constraints, which describe and limit certain aspects of this action. Each executor or target is represented by a set of attributes. Comparing to our previous works, we define an entity which is a set of related attributes. Executors and targets are subset of entities. Executors and targets are different roles that an entity can play in a policy, and these roles can change in different policies.

Policy-based management systems usually provide flexibility of constructing complex policies, which allow users to define multiple actions in one policy file. Therefore, an entity can be both executor and target in one policy file. In order to simplify the problem, we divide long policy into small segments. We assume that one segment describes one complete action containing one executor, one target, and its context (in the form of constraints). Therefore, a policy

segment is represented as a tetrad (Executor, Target, Action, and Context). In our policy model, a policy segment is the smallest functional unit. In order to present these components in logic expressions, we formally define several key elements in our policy model below. Table 1 describes these components of a policy.

Attribute (α)	A piece of information describing certain aspect of an entity.
Entity(E) = {α}	A collection of attributes describing a complete element in an information domain.
Executor ={α}	Executor is the entity that performs the action on another entity.
Target= {a}.	Target is the entity that receives the action from another entity.
Actioin()=Executor×Target.	A process of an entity affecting another entity or itself. An action is a relation between an executor and a target.
Context	Context in a policy segment includes all constraints on actions and entities.
Segment = (Executor, Target, Action(), Context)	The smallest functional policy unit in a policy.

Table 1. General Policy Model

We can use temporal logic to express policies in the examples from the introduction section:

Policy 1:

{A, D, setinterval(a),

[HoldsAt(permit(A,D,setinterval(a)),t)=true

iff A ∈DomainA∧D ∈DomainA ∧10ms<a<100ms] }

For "Policy 1", we add time information into the constraint without modifying the format of general policy model. Then the logical expression contains a temporal dimension denoted as "*t*". In this expression, if "*A"* and "*D*" belong to "*DomainA"* at time "*t*" and "*10ms<a<100ms*", the predicate "*permit()"* holds true.

Policy 2:

{B, D, SetDataErrorRate (1/100),

[HoldsAt(permit(B,D, SetDataErrorRate(100)),t)=true

iff B ∈DomainB∧D ∈DomainB∧(Dataerror$_D$<Dataerror$_B$=1/100)]}

In this expression, if "*B*" and "*D*" belong to "*DomainB*" at time "*t*", the predicate "*permit()"* holds true.

It seems that there is no conflict between these two policies. However, if we have a close look at the environment of this collaboration, we can find out that participant "*A*" cannot always increase its data rate requirement. Otherwise, sensor "*D*" cannot guarantee data quality for participant "*B*"'s need. In this situation, these two policies are conflicting with each other. Although human administrators can examine policies manually and may find out this possible conflict, extant policy analysis systems cannot go that far. They need extra information to detect possible conflicts. So we introduce a semantic extension containing domain and environment information to support logical reasoning.

C. Combination of Temporal Logic and Knowledge Base

In order to analyze properties that will change over time, we build a logical agent for policy conflict analysis using temporal logic. Among four typical parts (inference engine, knowledge base, sensor, and actuator), we extend the knowledge base with semantic extension. A semantic extension contains attributes, relationships, and dynamic constraints among attributes and relationships from an information domain. In an information domain, there is information not only changing along with others but also changing over time. This dynamic information imposes complication on logical analysis. And a traditional knowledge base is not enough for logical reasoning in policy conflict analysis with dynamic attributes and relationships. We propose a semantic extension to represent various types of dynamic information to support necessary logical reasoning for policy conflict analysis. Semantic extension is a formal representation of related information abstracted from an information domain. Related information contains attributes, entities, relationships and constraints in the information domain. Relationship is an important definition for the semantic extension. A relationship represents a connection between two entities.

Definition 1: Relationship (Θ) is represented by the Cartesian product of two entities: $\Theta=E\times E'$.

If Θ is a relationship between "*Participant A"* and "*Sensor D"*, then Θ is an instance of $A\times D$.

$\Theta=\{(\alpha, \beta)| \alpha \in A \text{ AND } \beta \in D\}$

In collaborative workspace, relationships are implied in policies to control information sharing. A relationship between two entities is a relationship between two attributes in different entities. However, there are attributes affecting relationships directly, and some are affecting indirectly. There are also some constraints acting on relationships.

Definition 2: Explicit attribute is an attribute that cause a change of another attribute through a relationship.

The superscript in a logical expression denotes an explicit attribute.

Definition 3: Implicit attribute is an attribute that is influenced by a change of other attributes through a relationship.

Implicit attributes are denoted as suffixes.

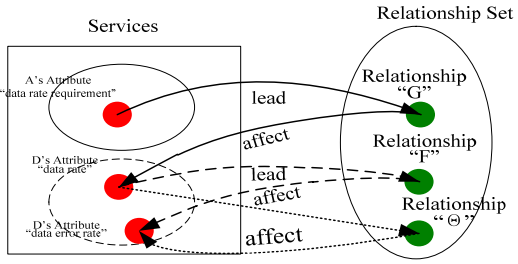

Figure 1 Relationships and Entities

The Figure 1 shows the difference between explicit and implicit attribute. In Figure 1, if explicit attribute "*data rate requirement*" changes, this change will lead relationship "*G*" to change and then the implicit attribute *"data error rate"* in D should also change. Because "*data error rate"* is an explicit attribute for relationship "*F*", the relationship "*F*" will also change if attribute "*data error rate*" changes.

We use semantic extension to represent these explicit and implicit attributes and track their updates.

$$\forall t.T<t \wedge HoldsAt(increase(A.data\_rate),t)$$
$$\wedge(10<A.data\_rate<100)$$
$$\wedge HoldsAt(G^{A.data\_rate}_{D.error\_rate},t) \wedge HoldsAt(F^{D.error\_rate},t)$$
$$\Rightarrow HoldsAt(permit(increase(A.data\_rate),t)$$

In this expression, if action "*increase throughput*" is hold, and requested throughput less than 90, and relationship "*G*" and "*F*" are hold, then the action "*increase()*" is permitted. We donate the attribute "*A.data_rate*", which is an explicit attribute for relationship "*G*", as a superscript, and the implicit attribute "*D.error_rate*" as a suffix.

A relationship does not only affect by attributes, but also affect by constraints. In an information domain, relationships connect different entities (participants and sensors). Because an entity is a set of attributes, relationships connect different attributes. Figure 3 illustrates this situation. In this figure, Participant (A and B) and sensor D have two relationships (policies). Participant (A and B) and sensor D have 3 attributes respectively. Relationship G connects attribute "data rate requirement" in Participant A and sensor D. If there is a constraint on attribute "data rate requirement" in Participant A, Relationship G will be affected only when this constraint (10ms<data rate<100ms in this case) is satisfied. If this constraint changes over time, we call this constraint dynamic constraint. Dynamic constraints are very important in an information domain, because these constraints control the connection between different entities. In addition, attribute "data rate requirement" is an explicit attribute for Relationship G, attribute "data error rate" is an implicit attribute for Relationship G, and attribute "data error rate" is also an explicit attribute for Relationship F. Therefore, if Attribute "data rate requirement" changes, it will affect Relationship G, and then Relationship G affects Attribute "data error rate", and Attribute "data error rate" affects Relationship F finally. If results of these two changes are inconsistent, there will be a conflict (conflict of duty).

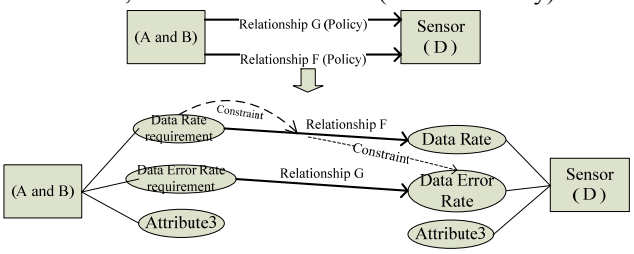

Figure 3 Relationships and constraints

Relationships in an information domain are connections between entities, which are sets of attributes (Figure 3). Therefore, relationships are connections of attributes.

A relationship is:

R=data_rate_requirement×data_error_rate_requirment,

because there is also a constraint limits this relationship, we have to consider this constraint during logic reasoning. If we consider this constraint, the relationship becomes two sub-relationships: R'(c) and R-R'(c) (Figure 4 b). R'(c) is means values of relationship when constraint "c" is true or becomes effective, R-R'(c) means values of relationship that are not affected by constraint "c". However, in the semantic extension, we consider these two sub-relationships as a complete relationship, which can be express as R=(R-R'(c))∪R'(c) (Figure 4 c). Sometime, constraints are not active, the relationship become: R=(R-R'(c))∪R'(c) ⇒ R=(R-ϕ)∪ϕ=R.

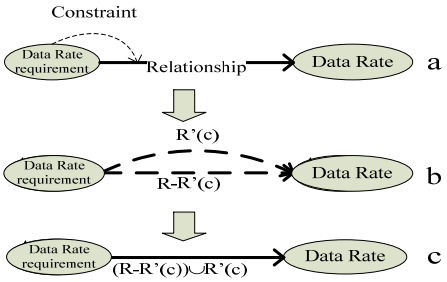

Figure 4 Relationship and Constraints

Definition 4: Constraint (*Δ*) is restrictive information on attributes or relationships.

Relationships can be expressed as "$\Theta=\chi\times\upsilon$", where "$\chi$" and "$\upsilon$" are attributes in different entities, and "Θ" is a relationship. A constrained relationship becomes "$\Theta'(\Delta)=(\chi\times\upsilon)\times\Delta$", where "$\Delta$" is one or a set of constraints. In semantic extension, constraint "Δ" is a predicate that returns whether the constraint is satisfied or not. Constraints are conditions that restrict changes of attributes and relationships. Only when constraints are satisfied, an attribute or relationship can change to a certain value.

A semantic extension abstracts certain information from an information domain. Now we can give a definition for semantic extension.

Definition 5: A semantic extension contains attributes, entities, relationships and constraints from one information domain. $\Sigma=\{\{\alpha\},\{E\},\{\Theta\},\{\Delta\}|\ E\subseteq\{\alpha\},E\neq\varnothing,\Theta=E\times E'\}$.

Attributes in an information domain are not only associated with entities in the domain but also attributes describing properties of the domain. These are domain attributes that do not constitute entities usually. However, domain attributes may be added to entities in some situation. For example, when two semantic extensions merge together, the attribute "*domain ID*" may become an attribute of an entity.

Usually we use one semantic extension to present one information domain. One knowledge base can contain more

than one semantic extension, which depends upon the scope of this knowledge base. In a semantic extension, values of relationships are associated with corresponding attributes. When constraints on these relationships are satisfied, changes of attributes will affect relationships. Because some constraints will change over time, we use temporal logic to represent these dynamic constraints.

For example (as illustrated in Figure 3), relationship "*F*", which connects service "*A*", "*B*" and "*D*", has one explicit attribute "*Data Rate Requirement*" in service "*A*" and one implicit attribute "*Data Rate*" in service "*D*". And there is a constraint "*Δ*" on "*Data Rate Requirement*" and "*F*". Relationship "*G*" connects one attribute "Data Error Rate Requirement" in service "*B*" and one attribute "*Data Error Rate*" in service "*D*". Relationship "*Θ*" connects attributes "*Data Rate*" and "*Data Error Rate*" in service "*D*". When constraints are satisfied, "*Data Rate Requirement*" will affect "*F*", and "*Data rate*" in service "*D*". And this change is transferred through relationship "*Θ*" and affects "*Data Error Rate*" in service "*D*". Then the change of "*Data Error Rate*" will change the relationship "*G*". This situation can be represented as follows:

$$\forall t,(T<t) \wedge HoldsAt(F^{A.DataRateRequirement}_{D.DataRate}=Equal,t)$$
$$\wedge HoldsAt(\Theta^{D.DataRate}_{D.DataErrorRate}=Balance,t)$$
$$\wedge HoldsAt(G^{D.DataErrorRate}_{.B.DataErrorRateRequriement}=Larger,t)$$
$$\wedge(\text{HoldsAt}(\Delta,T)) \wedge Change(A.DataRateRequirement,t)$$
$$\Rightarrow Change(D.DataRate,t) \wedge Change(D.DataErrorRate,t)$$
$$\wedge Change(\Theta,t)$$

In this logical expression, constraint "*Δ*" holds after time "*T''*. Therefore, after time "*T''*, if explicit attribute "*DataRateRequirement*" in service "*A*" changes, implicit attribute "*DataRate*" in service "*D*" will change; attribute "*DataErrorRate*" in service "*D*" will change because of relationship "*Θ*"; attribute "DataErrorRate" in service "*D*" will affect relationship "*G*".

IV. Agent Architecture of Conflict Analysis Engine

A conflict analysis agent is an autonomous entity which observes environment and acts upon the environment. In this paper, we propose conflict-analysis agent architecture (Figure 5) for conflict analysis using temporal logic and semantic extension. In the inference engine of the agent, the first component is "policy decomposition". This component decomposes policies into subjects, objects and other policy components according to the role information and domain information from the knowledge base, especially from the semantic extension. The next component is query module. This component queries semantic extension to get relationship information between different policy elements. The temporal logic module uses analysis rules from knowledge base to analyze policies, and forward analysis result to the reconciliation module. The reconciliation and suggestion module provides suggestion for users.

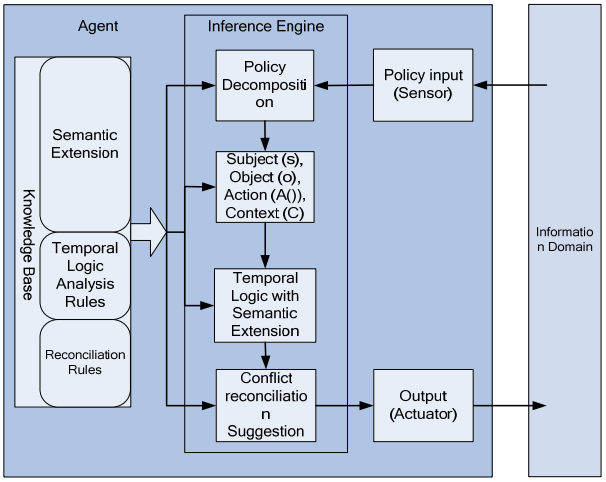

Figure 5 Conflict-analysis Agent Architecture

The knowledge base in this architecture consists of semantic extension, temporal logic analysis rules, reconciliation rules and other domain information. The semantic extension contains formal representations of relationships between attributes, constraints on relationships and other related information from an information domain. Temporal logic analysis rules contain conflict detection rules. We will discuss certain common rules in the next section. Reconciliation rules are established according to different types of conflicts. These rules provide suggestions for users to solve corresponding conflicts.

V. Automatic Conflict Analysis in Collaborative Workspace

A. Workspace Collaboration Environment

Recent technological advancements have made deployment of small, inexpensive, low-power, distributed devices, which are capable of local processing and wireless communication, a reality. In most workspace there are lots of sensors, some are monitoring temperature, some are monitoring water level. Sensor node is a representative device. Each sensor node is only capable of doing a limited data processing. However, networks of these small sensor nodes can monitor many aspects of our daily lives. Early sensor networks consist of a small number of sensor nodes that are wired to a central processing station. Nowadays, sensor networks become distributed and wireless, and functions of sensor networks become more diverse. Collaborations between sensor nodes and between sensor node and other participants have developed. For example, a sensor monitors the water level of a water tank, where two participants ("*A*" and "*B*") reading data from this sensor. This sensor provides limited computational power for increasing the quality of its data. Participant "*A*" generates flood alarms for monitoring agencies; participant "*B*" records water level periodically as records. If the water level does not exceed a certain threshold, "*A*" only receives data from the sensor. However, if the water level exceeds this threshold,

"*A*" will request sensor data more frequently. At this time point, the sensor has to consume more power for sending data and has to reduce its data quality. How to configure this type of sensor nodes for accommodating collaboration is a major issue for sensor network collaboration. One policy presents one configuration of a sensor node. If one policy conflicts with another policy, this may cause a sensor failure or data failure. Our proposed knowledge-augmented logical agent can be applied here to detect such conflicts in sensor networks environment.

According to our knowledge-augmented logical framework, these policies can be expressed as the following logical expressions.

Policy 1:

HoldsAt(permit(A,D,setinterval(a)),t)=true iff 10<a<100

Policy 2:

HoldsAt(permit(B,D, SetDataErrorRate (1/100)),t)=true

In the above logical expressions, there is no conflict between these two policies. In Figure 6, the sensor's interval for participant "*A*" will change between 10s and 100s; the sensor's data quality for participant "*B*" will be constant and higher than "*B*"'s requested level. These two policies define over different attributes in the sensor.

However, if we analyze the entire collaboration, we find that participant "*A*" cannot always increase its requirement over data acquisition frequency. Otherwise, the sensor will not be able to provide enough data quality to participant B. Figure 7 illustrates the correlation of these two policies. In this diagram, when water level increases, the data acquisition frequency needed by participant "*A*" will also increase. So the sensor has to use more computational resources for participant "*A*". At time "*t*", the data error rate of sensor's output meets the requirement of participant "*B*". At time "*t'*", the data error rate of sensor's output cannot meet the requirement of participant "*B*". If participant "*A*" continues to increase its requirement, the sensor cannot provide qualified data for participant "*B*". In this situation, these two policies are conflicting with each other. Although human users can sometimes find this implicit conflict manually, computer cannot assess this implicit impact by itself. It needs someone to provide extra information. So the proposed semantic extension that contains domain or environment information is necessary for logical reasoning upon implicit relationships and resultant conflicts.

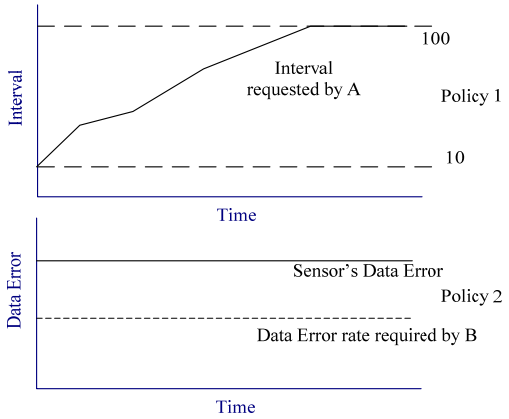

Figure 6 Time Interval and Data Quality in Two Policies

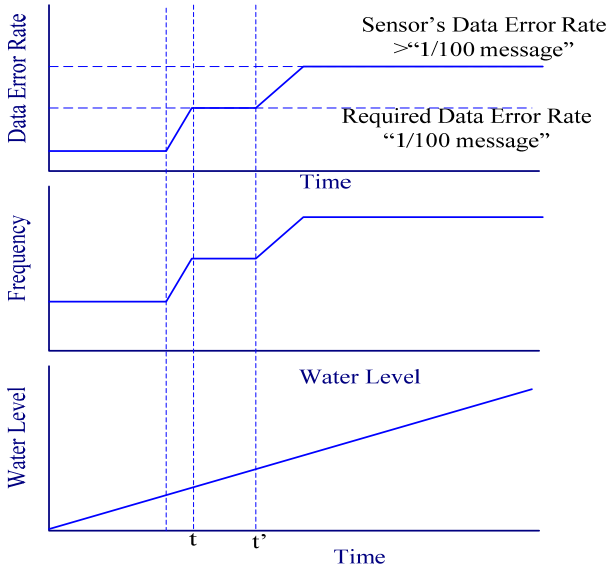

Figure 7 Relationship between Two Policies

B. Implementation rules

In this case study, relationships among two participants are also combined with logical expressions to represent their policies. Here we just illustrate one example implementation rule.

$$\forall t.(T<t) \wedge HoldsAt(permit(A,Sensor, setinterval(A,Sensor)),T) \wedge HoldsAt(permit(B,Sensor, setdataerrorrate(1/100)),T) \wedge HoldsAt(\Phi^{A.requestedfrequency}_{Sensor.frequency}=Equal,t) \wedge HoldsAt(\Theta^{B.dataerrorrate}_{Sensor.dataerrorrate}=Smaller,t) \wedge HoldsAt(IT^{Sensor.frequency}_{Sensor.dataerrorrate}=Balance,t) \Rightarrow HoldsAt(dynamicConflict(ConflictofDifexecutor,overlaps (permit(A,Sensor,setinterval(A,Sensor)), permit(B, Sensor, setdataerrorrate (1/100)),t) \Rightarrow Trajectory(permit(B,Sensor,setinterval (B,Sensor)),T,$$

deny(B, Sensor, setdataerrorrate (1/100)),t)

[Situation: The executor ("A") has a "level" attribute. This attribute can change over time. This attribute is an explicit attribute for a relationship *"Φ"*. This relationship allows the executor to perform action "change data frequency of sensor". Attribute "*Sensor.frequency*" is an implicit attribute for this relationship *"Φ"*. Attribute "*B.dataerrorrate*" is an explicit attribute for relationship *"Θ"*. The relationship *"Θ"* keeps the sensor providing qualified data to outside participants. This information is stored in semantic extension. If the attribute "level" changes, the executor will also change. Then an overlap conflict will occur.]

C. *Experiments*

In the experiments for this case study, we choose three set of policies A, B, C. These policies come from two different sensor systems. We copy these policies before these two sensor systems collaborate. 30 policies are from one sensor system, and the other 30 are from the other system. There are 20 policy pairs in each set. There are 15 static conflicts in set A, 16 dynamic conflicts in set B, and 13 dynamic conflicts in set C. Conflicts in policy set B are dynamic conflicts but there is no explicit attribute or implicit attribute involved in any conflict. Conflicts in policy set C are also dynamic conflicts, but certain explicit and implicit attributes are involved in conflicts. We use temporal logic and temporal logic with semantic extension to analyze each policy set. In set A, two logics report the same accuracy; in set B, there is no difference in the analysis result too; while the third set shows some difference. The analysis result of the temporal logic is not accurate. There are only 8 reported conflicts, which means 5 conflicts are not reported. The temporal logic with semantic extension reports all 13 conflicts. The result shows that if a conflict is caused by implicit relationships and constraints on relationships, the temporal logic with the support from an extended knowledge base can provide more accurate result than pure temporal logic.

The Figure 8 shows the result of this experiment. In the first two policy sets, the analysis result from temporal logic and temporal logic with semantic extension are the same, because these conflicts are caused by executors and targets themselves, and there is no transitional relationship or implicit attribute within a relationship. In policy set C, conflicts are caused by implicit attributes and transitional relationships. Different results illustrate how relationships and implicit attributes affect analysis result. If implicit attributes appear in a relationship, pure temporal logic won't consider the transitivity of these attributes. This transitivity is achieved through the relationship(s) among entities or attributes. Only knowledge base containing this transitivity information can help detect conflicts caused by these attributes.

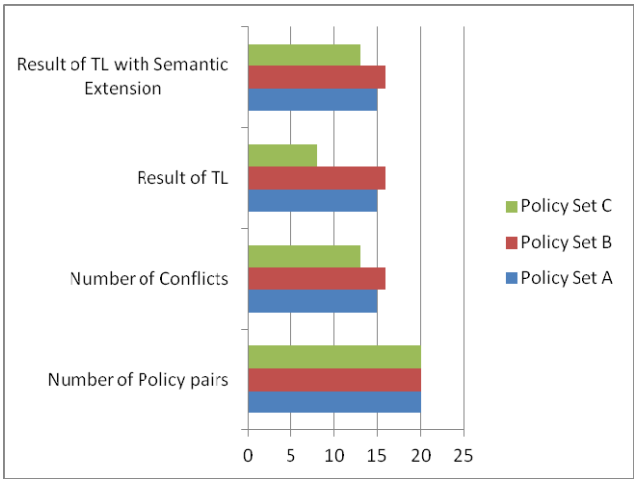

Figure 8. Comparison of Analysis Results in Sensor Network Environment

VI. DISCUSSION

Changing environment and entities are big challenges for policy analysis, because dynamic information and relationships are hard to represent and analyze during logical reasoning. Temporal logics are widely used in conflict analysis. However, the dynamic relationship is still a barrier for logical reasoning. Most previous systems rely on human interference. In extant approaches, relationship and constraint information is mentioned, but not integrated into logical reasoning. In [3], semantic information is specified by means of laws and constraints. These laws and constraints cannot be modified during the analysis. If any constraint is changed, users have to change reasoning rules. In [4], a logical reasoning framework is presented, and authors also presents a specification language FCTL (First Order CTL) for specifying properties in trust management system. The framework is designed for dynamic policies. However, authors transfer policies into a time-bounded format. This only works in a well-known environment, because if there is a dynamic attribute affected by environment factors, this approach will not respond promptly. In [5], authors propose a temporal logic with temporal constraints, in which a temporal logic (Fuzzy Temporal Logic, FTL) is used to support efficient query answering. However, this temporal logic with temporal constraints is too simple in terms of its fuzzification method. This method only uses an interval as a metric constraint. If an information domain is very complex, this temporal logic cannot support semantic domain information. In [6], a generalized temporal logic is introduced. It includes two extensions: CTL and the μ-calculus. Both are defined over an algebraic structure (c-semirings) capturing many soft constraint satisfaction problems (CSP). It is difficult for these extant approaches to work in complex information domains, because relationships and dynamic contexts cannot be captured and represented and will eventually reduce the accuracy of logical reasoning. In our proposed knowledge-augmented temporal logic, the semantic extension can convey these relationships

(sometimes even implicit relationships or constraints) for a complex information domain, so the semantic extension can ensure the accuracy of logical reasoning.

Policy analysis includes static analysis and dynamic analysis. Both ACLP [9] and an event-driven model [15] can monitor run-time policies. These frameworks concentrate on static policy set. If policy set is changed, they have to re-analyze the entire policy set. So these systems can only detect static conflicts instead of dynamic conflicts. The event-driven mechanism [15] uses a conflict database to store all possible conflicts. The capability of this approach is also limited by this reliance on conflict database. Dynamic analysis gains more attention in recent years. Several dynamic analysis systems will be discussed and compared here. In [16], authors implement Event Calculus in a QoS management environment to analyze QoS management policies. This framework only works in a single domain. In [17], authors use Boolean rules and corresponding algorithms to discover and resolve two types of dynamic IPSec conflicts. One is runtime analysis; the other is dynamic information analysis. Dynamic information analysis focuses on dynamic elements in a domain, which may change over time. Temporal logics can represent time related information and analyze this type of information. In [16], authors propose a policy analysis framework using event calculus to analyze policies from a single domain. Although this framework works on dynamic information, it only works for one single domain. It cannot deal with policies from multiple domains, because the same attribute may have different names or definitions in different domains, which may cause ambiguity. And the complexity of integrating different domain information is also a barrier for multiple domain policy analysis. In the proposed knowledge-augmented temporal logic, semantic extension supplies information from multiple domains and also relationships between entities. It can reduce ambiguity, which usually happens in multi-system integrations.

VII. CONCLUSION

Temporal logics have been study for decades, several techniques have been developed. The logic representation and reasoning functionalities are used in conflict analysis area. In collaborative environments, when temporal logics are used for analysis, logic reasoning is affected by domain information. In this paper, we integrate temporal logic with a semantic extension, which contains information of an information domain. Through the experiments on our prototype system, the improvement on capability and accuracy of combination of knowledge base and temporal logic for automatic policy analysis is confirmed. And it also reduces human interventions. The knowledge base is flexible for adapt to dynamic collaboration and system integration.

REFERENCES

[1] A. Westerinen, J. Schnizlein, "RFC3198 - Terminology for Policy-Based Management", 2001 http://www.faqs.org/rfcs/rfc3198.html

[2] Jesper G. Henriksen, P. S. Thiagarajan, "Dynamic Liner Time Temporal Logic", Annals of Pure and Applied Logic, Volume 96(1-3), 1 1999, pp. 187-207

[3] Laura Giordano, Alberto Martelli, Camilla Schwind, "Specifying and verifying interaction protocols in a temporal action logic", *Journal of Applied Logic*, Vol.5(12), 2007, pp. 214-234.

[4] A. Prasad Sistla, Min Zhou, "Analysis of dynamic policies", *Information and Computation,* Vol. 204 Issue 2-4, 2008, pp. 185-212.

[5] Liguo Deng ; Yunpeng Cai ; Chen Wang ; Yan Jiang, "Fuzzy Temporal Logic on Fuzzy Temporal Constraint Networks", *Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, Vol. 6, 2009, pp. 272 – 276.

[6] Alberto Lluch-Lafuente, Ugo Montanari, "Quantitative μ-calculus and CTL Based on Constraint Semirings", *Electronic Notes in Theoretical Computer Science*, Vol. 112, 2005, pp. 37-59

[7] Arun K. Eamani, A. Prasad Sistla, "Language based policy analysis in a SPKI Trust Management System," *Journal of Computer Security*, Vol. 14(4), 2006, pp. 327-357.

[8] Gail-Joon Ahn, Wenjuan Xu, Xinwen Zhang, "Systematic Policy Analysis for High-assurance Services in SELinux", *Proceedings IEEE Workshop on Policies for Distributed Systems and Networks,* 2008. pp.3-10.

[9] Robert Craven, Jorge Lobo, Jiefei Ma, "Expressive Policy Analysis with Enhanced System Dynamicity," *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security,* 2009, pp. 239-250.

[10] Vladimir Kolovski, James Hendler, Bijan Parsia, "Analyzing Web Access Control Policies", *Proceedings of the 16th international conference on World Wide Web,* 2007, pp. 677–686.

[11] Patrick McDaniel and Atul Prakash, "Methods and limitations of security policy reconciliation," *ACM Transactions on Information and System Security,* Vol. 9, No. 3, 2006, pp.259-291.

[12] M. Blaze, J. Feigenbaum, and Jack Lacy, "Decentralized Trust Management," *Proceedings of 1996 IEEE Symposium on Security and Privacy,* 1996, pp. 164 -173.

[13] Tatyana Ryutov, Clifford Neuman, "The Specification and Enforcement of Advanced security Policies," *Proceedings of the 2002 Conference on Policies for Distributed Systems and Networks*, 2002, pp.0128.

[14] Zhengping Wu, Yuanyao Liu, "Knowledge-Based Policy Conflict Analysis in Mobile Social Networks," *Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, 2011, pp.1-6.

[15] N. Dunlop, J. Indulska, K. Raymond, "Dynamic conflict detection in policy-based management systems," *Proceedings of Sixth International Enterprise Distributed Object Computing Conference*, 2002, pp. 15- 26.

[16] Charalambides, M. Flegkas, et al., "Policy conflict analysis for diffserv quality of service management," *IEEE Transactions on Network and Service Management*, vol.6 (1), 2009, pp.15-30.

[17] S. Niksefat, M. Sabaei, "Efficient Algorithms for Dynamic Detection and Resolution of IPSec/VPN Security Policy Conflicts," *Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 2011, pp.737-744.