SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

January 2012 Vol. 55 No. 1: 149–164
doi: 10.1007/s11432-011-4426-1

A novel logic-based automatic approach to constructing compliant security policies

BAO YiBao[1,2,4], YIN LiHua[1]*, FANG BinXing [1,3] & GUO Li [1]

[1] *Institute of Computing Technology, Chinese Academy of Sciences,*
Beijing 100190, *China;*
[2] *Institute of Electronic Technology, Information Engineering University,*
Zhengzhou 450004, *China;*
[3] *Beijing University of Posts and Telecommunications,*
Beijing 100190, *China;*
[4] *Graduate University, the Chinese Academy of Science, Beijing* 100049, *China*

Received September 1, 2010; accepted April 14, 2011; published online November 26, 2011

Abstract It is significant to automatically detect and resolve the incompliance in security policy. Most existing works in this field focus on compliance verification, and few of them provide approaches to automatically correct the incompliant security policies. This paper proposes a novel approach to automatically transform a given security policy into a compliant one. Given security policy Π and delegation policy M declared by logic programs, the approach automatically rewrites Π into a new one Π_M which is compliant with M and is readable by the humans. We prove that the algorithm is sound and complete under noninterference assumption. Formally, we show that the security policy query evaluation algorithm with conflict and unsettlement resolution still works very well on Π_M. The approach is automatic, so it doesn't require a administrator with excess abilities. In this sense, our proposal can help us to save much manpower resource in security management and improves the security assurance abilities.

Keywords security policy, rewriting, logic program, compliance

Citation Bao Y B, Yin L H, Fang B X, et al. A novel logic-based automatic approach to constructing compliant security policies. Sci China Inf Sci, 2012, 55: 149–164, doi: 10.1007/s11432-011-4426-1

1 Introduction

In practice, in a security policy management system (SPMS), a security manager (i.e., delegator) normally breaks a complex security policy management task down into several manageable chunks. Then the manager delegates them to several other administrators (i.e., delegatees), and hopes that the security policies submitted by the delegatees should be compliant with a set of delegation restrictions (i.e., delegation policy) pre-defined by the delegator. If not so, the delegator hopes that there is a mechanism that can help him to automatically manage the compliance between the security policies and the pre-defined delegation policy. Furthermore, when any incompliance is found, the delegator hopes that there is an approach that can automatically resolve the incompliance and compulsively impose the delegation restrictions on the submitted security polices. This is called compliance maintenance in this paper. If an

---

*Corresponding author (email: yinlihua@software.ict.ac.cn)

dr_1	A can only grant the users designated with administrative role (i.e., the administrators) to browse the *core* module.
dr_2	A can grant all users to browse the *news* module.
dr_3	The administrators except super ones who are not temporarily suspended from duty can not designate or revoke the administrative role to or from any users.

Figure 1 Delegation policy 1 (declared in natural language).

pr_1	Any users who are trusted by A are allowed to browse *core*.
pr_2	A designates the administrative role to user B.
pr_3	A trusts C and D.

Figure 2 Security policy 1 (declared in natural language).

SPMS can support this, its task on security policy management would be more automatic, easier, and more manageable, and the security policies generated from the SPMS should be more secure.

Informally, given a delegation policy M and a security policy Π, Π is compliant with M, namely, if Π does not violate the delegation restrictions defined by M. Intuitively, compliance depicts the inherent constraint relationship between delegation policies and security policies, such as delegation policy 1 shown in Figure 1 and security policy 1 shown in Figure 2.

Compliance maintenance has an important role in SPMS. For example, there is a information system SecApp whose partial security policy management task is delegated to an administrator A. Suppose SecApp is guarded by an RBAC [1] security policy and its delegation policy is shown in Figure 1, where *core* and *news* are two sub-modules of SecApp.

When A writes a security policy shown in Figure 2 and submits it to SecApp, for the sake of security, SecApp should check whether security policy 1 is compliant with delegation policy 1. We can see that it is not so, but how an SPMS can “see” or resolve this? Currently, when security policy 1 is submitted to SecApp, SecApp has two typical alternatives to keep it compliant with delegation policy 1:

Alternative 1: Monitoring the policy management actions [2] under a reference monitor, which is illustrated in Figure 3(a) and its philosophy is that the delegation policy is treated as an access control policy which is used to guard the actions that are used to create or modify another security policy.

Alternative 2: Verifying and validating the security policy with its intended delegation policy [3, 4]. If it is compliant with M, accept it. If not so, reject it. This is illustrated in Figure 3(b) and its philosophy is that the delegation restricts are treated as security properties of security policy to be verified and validated.

With Alternative 1, on the one hand, it is so difficult to design and implement a sound and complete reference monitor [5, 6] that many practical SPMSs based on this framework are the results of the compromise between their implementations and their soundness and completeness. As a result, when a security policy is submitted to an information system, as shown in Figure 3(a), it must be verified and validated before it becomes applicable. On the other hand, the reference monitors only can monitor the actions to create the security policies, but can not analysis their semantics. Obviously, delegation policy 1 is declared on the semantics of its intended security policies, but not on the actions to create the security policies. Obviously, Alternative 1 can not be applied in this scenario.

Most of the related works [3,4,7–9] focus on the second alternative and several generalized theories and approaches have been proposed, such as software verification and validation [10], model checking [11, 12]. However, these approaches have the following shortcomings: (1) Constructing a verifiable security policy is out of an administrator’s ability. Since policy verification and validation is so difficult that only strictly trained expertise can use these approaches. (2) These approaches can only be used to verify the security policies, while they could not automatically resolve the found incompliance. Usually, the administrators will take lots of time to manually resolve it, which is time-consuming and boring.

There are some scenarios that Alternative 2 can not be enforced. For example, suppose there is an information system BlpApp which is guarded by BLP security policy [13] with $\{0, 1, 2, 3\}$ as its security

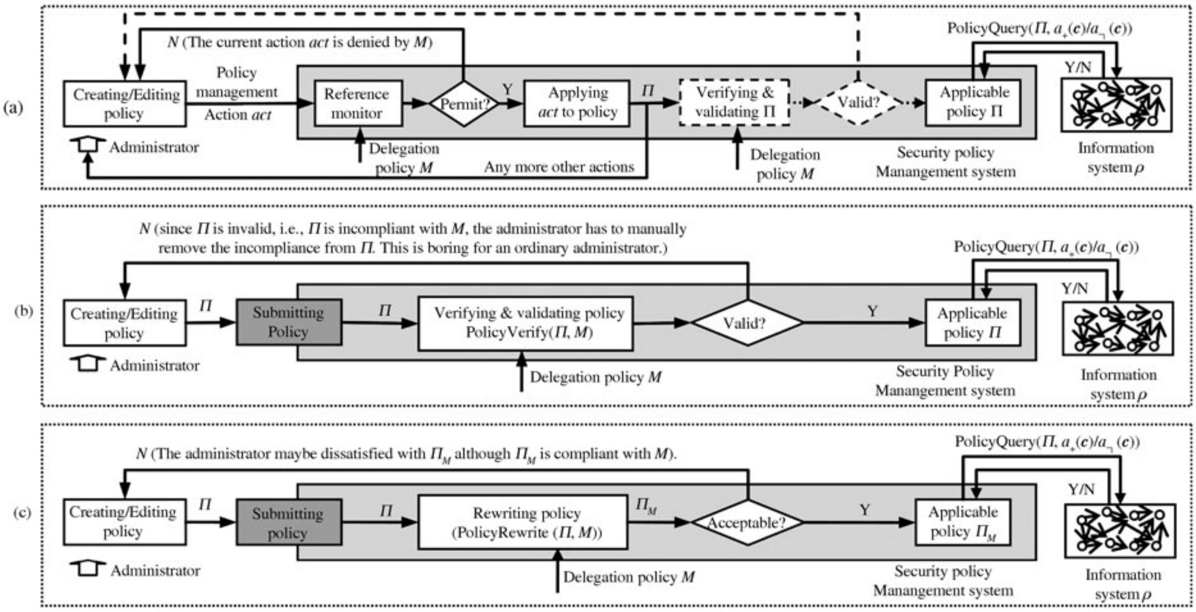
(a)
N (The current action act is denied by M)
PolicyQuery(Π, a₊(c)/a¬(c))
Y/N
Creating/Editing policy
Policy management Action act
Reference monitor
Permit?
Y
Applying act to policy
Π
Verifying & validating Π
Valid?
Applicable policy Π
Administrator
Delegation policy M
Any more other actions
Delegation policy M
Security policy Manangement system
Information system ρ
(b)
N (since Π is invalid, i.e., Π is incompliant with M, the administrator has to manually remove the incompliance from Π. This is boring for an ordinary administrator.)
PolicyQuery(Π, a₊(c)/a¬(c))
Y/N
Creating/Editing policy
Π
Submitting Policy
Π
Verifying & validating policy PolicyVerify(Π, M)
Valid?
Y
Applicable policy Π
Administrator
Delegation policy M
Security Policy Manangement system
Information system ρ
(c)
N (The administrator maybe dissatisfied with Π_M although Π_M is compliant with M).
PolicyQuery(Π, a₊(c)/a¬(c))
Y/N
Creating/Editing policy
Π
Submitting policy
Π
Rewriting policy (PolicyRewrite (Π, M))
Π_M
Acceptable?
Y
Applicable policy Π_M
Administrator
Delegation policy M
Security policy Manangement system
Information system ρ

Figure 3 Suppose that the administrator wants to create a security policy Π for ρ.

dr_1	A is only allowed to label subjects and objects with security levels 0 or 1.
dr_2	B is only allowed to label subjects and objects with security levels 2 or 3.

Figure 4 Delegation policy 2 (declared in natural language).

pr_1	User u_1 is labeled by A with security level 1.
pr_2	Object o_1 is labeled by A with security level 2.
pr_3	If A scores the reputation of a user with X, then A labels him with security level X.
pr_4	A scores the reputation of user u_2 with 0, u_3 with 1 and u_4 with 3.
pr_5	If a use's reputation is not scored, A scores the user's reputation with 0.
pr_6	$u_1, \ldots, u_5$ are the users in the information system.

Figure 5 Security policy 2 (declared in natural language).

levels. BlpApp delegates its security policy maintenance tasks (e.g., labeling the security levels of objects and subjects) to an administrator A under the delegation policy shown in Figure 4. Suppose A writes a security policy shown in Figure 5. Obviously, delegation policy 2 is declared through restricting the security policy management actions. Hence, Alternative 2 can not be applied to this scenario.

Most of the current innovative security policy (management) frameworks [14–19] only focus on security policy expression or verification. Less works focus on the inadequacies existing in Alternative 1 or 2. In this paper, we propose the third alternative.

Alternative 3 (A new idea): Automatically rewrite a security policy Π into a new one Π_M which is inescapably compliant with its intended delegation policy M, and at the same time maximally reserves the valid part of the semantics of Π. This is illustrated in Figure 3(c) and its philosophy is that every delegation policy is treated as a special kind of security policy which defines the maximal set of privileges that the intended delegatees could authorize to the other subjects.

2 Overview of our approach

Security policy defines the security properties for a or a set of information systems. Practically, a security property are depicted by guarding the actions in an information system [2]. Based on this observation, we claim that a security policy is mainly used to define which actions are allowed or denied by an information system, while a delegation policy defines the set of allowed or denied actions that its intended security policies should follow. That's to say, if an action is denied by a delegation policy, it should also be denied by its intended security policy. Respectively, if an action is allowed by a security policy, it must be allowed by its intended delegation policy.

Generally, it is difficult to ensure that a security policy follows another delegation policy. Furthermore, we claim that it is a great challenge to attack Alternative 3, which could not be conquered in a short time. Hence, in this paper, we only discuss a simplified scenario, where security policy and delegation policy are declared with logic programs [20], and conflicts and unsettlements are resolved [15] from them, and both of them are noninterference (noninterference is defined in Definition 9).

Our approach is shown in Figure 3(c). A submitted security policy Π will be automatically disposed by a rewriting algorithm PolicyRewrite embedded in the SPMS. The rewriting algorithm will try to resolve any incompliance from Π with M (i.e., soundness), and maximally reserve Π's valid semantic to keep Π serviceable (i.e., completeness). This is the main difference of our approach from the others. Finally, the rewriting algorithm outputs a new, human readable security policy Π_M which is coded by the same language as Π. Hence, the administrator has the chances to review Π_M.

As we all know, the core of a delegation policy is to define the management privileges authorized to the delegatees for managing the security policies. In the narrow sense, many researchers think that management privileges are only the constraints on the management actions. Hence, they think that the policy shown in Figure 4 would be delegation policy, but not the one shown in Figure 1. Essentially, we claim that, whatever the expression is, a delegation policy defines a set of privileges that could be authorized to some other subjects by the delegatees. From this viewpoint, the policies shown in Figure 1 and Figure 4 are all delegation policies, but they are declared through two different kinds of strategies:

Eager strategy: SPMS directly monitors the whole security policy management process, such as Figure 3(a) shown. When an action is not allowed by the delegation policy, the reference monitor denies it. If the reference monitor is sound and complete, the generated security policies should be compliant with their intended delegation policies. Unfortunately, with this strategy, the delegators cannot abstractly declare their security goals in a high-level. Eager strategy is intuitive, but it is difficult to be formalized.

Lazy strategy: SPMS does not care about the actual policy management process. However, SPMS delays all the compliance maintenance task until the security policy is submitted, such as Figure 3(b) and 3(c) shown. Obviously, with this strategy, the delegators can abstractly declare their security goals in a high-level, and they needn't care about the implementation details of SPMS. Lazy strategy is simple and easy to be formalized. Actually, the existing works [14–18] bring us many hints to formalize an SPMS with lazy strategy. Since the SPMSs with lazy strategy activate the compliance maintenance process at the last stage, they are more efficient and easier to be used than those with eager strategy.

Eager strategy and lazy strategy have different philosophy. The former ensures its correctness through the correctness of the process. The latter ensures its correctness through the final disposition. Notice that the difference between them is a key to understanding the approach proposed in this paper. However, there are no insurmountable boundaries between them. For example, delegation policy 2, declared in eager strategy, can be easily re-declared in lazy strategy, shown in Figure 6. The above taxonomy shows that our approach, which will be discussed in detail in Section 4, is very efficient and easy to be used.

From the above discussions, we can see that our proposal is a new philosophy, and PolicyRewrite is a mechanism to implement the philosophy. In this new philosophy, the role of delegation policy is changed from defining the privileges authorized to the administrators to defining the privileges that the administrators can authorize to the other subjects. This has the following advantages: (1) Π_M is compliant with M and maximally reserves the serviceability of Π. We can avoid security policy verification, manual incompliance resolution, and reference monitor implementation; (2) For any administrator, if he is able to write Π, then he is able to make use of our proposal. Our approach doesn't require any excess abilities for the administrators; (3) Our proposal consumes less manpower and processor time than the others shown in Figure 3(a) and 3(b). In this sense, our proposal improves the security

dr'_1 :	The security levels of the subjects and objects labeled by A must be 0 or 1.
dr'_2 :	The security levels of the subjects and objects labeled by B must be 2 or 3.

Figure 6 Delegation policy 2 (declared in natural language with lazy strategy).

assurance abilities.

As we know, logic program [20] is an excellent tool for constructing policy languages and frameworks [14–19]. Our proposal is also logic program based and it can be easily integrated into the existed works. For conveniences, we assume in this paper that M and $\varPi$ are always two logic programs, where $\varPi$ is a security policy with M as its intended delegation policy.

The rest of this paper is organized as follow: In Section 3, we devote to model the policy-based information system. Based on the model, we introduce a logic program based security policy framework used in our approach, which includes policy language, policy semantic, conflict detection and resolution, unsettlement detection and resolution, and policy query evaluation. In Section 4, given M and $\varPi$ declared in the above policy framework, we propose an algorithm PolicyRewrite$(\varPi, M, \varPi_M)$ to automatically rewrite $\varPi$ into a compliant security policy $\varPi_M$ with M. We prove that the algorithm is sound and complete under the semantics decided by the query evaluation algorithm introduced in Section 3. This is our main contribution. In Section 5, we give a simple example to depict how to use our approach. We conclude this paper in Section 6.

3 Policy-based information system

In this section, we will give a formal model of policy-based information system and introduce some technical terms. Then we propose a policy framework to formally declare the delegation policy and security policy. We suppose the readers are familiar with first-order logic. Let p be any n-arity predicate. Then $p(\boldsymbol{t})$ is an n-arity atom, where $\boldsymbol{t}$ is a sequential n-tuple of constants (usually denoted by string beginning with lower-case, such as a, b, c) or variables (usually denoted by string beginning with upper-case, such as X, Y, Z). If $p(\boldsymbol{t})$ is an atom, then $p(\boldsymbol{t})$ and *not* $p(\boldsymbol{t})$ are literals, where the former is positive and the latter is negative. Especially, if every tuple of $\boldsymbol{t}$ is constant, $p(\boldsymbol{t})$ is ground, and $p(\boldsymbol{t})$/*not* $p(\boldsymbol{t})$ are positive/negative ground literal.

3.1 Modeling the information systems

In order to formally show the role of security policy and delegation policy in an information system and the relationship between them, first, we have to accurately specify what an information system is.

Definition 1 (Information system). Formally, an information system is represented by a 6-tuple $\rho(Q, Q_0, \mathcal{E}, P, M, \varPi)$. ρ is an unique identity of the information system; Q is a set of states; Q_0 is a set of initial states, and $Q_0 \subseteq Q$; $\mathcal{E}$ is a set of events which transfer the system from one state to another. Every event $\varepsilon \in \mathcal{E}$ is a function $\varepsilon : Q \to Q$. Practically, every event $\varepsilon(q, q')$ can be explicitly expressed as $\varepsilon(\boldsymbol{c})$, where $\boldsymbol{c}$ is a sequence of any input or output constant arguments with fixed arity. Hence each event can be treated as a special kind of predicate. Each n-arity event ε introduces two new n-arity predicates ε_+ and ε_- into $\mathcal{E}$, called event predicates. Event atom $\varepsilon_+(\boldsymbol{t})$ (resp. $\varepsilon_-(\boldsymbol{t})$) means that event $\varepsilon(\boldsymbol{t})$ is allowed (resp. denied) to be executed on argument sequence $\boldsymbol{t}$. P is a set of predicates used to depict the relationships among the entities in the system. P is divided into two disjoint sets $\mathcal{E}$ and $\varOmega$. $\mathcal{E}$ is described above. The predicates in $\varOmega$ are called characteristic predicates which are used to depict the characteristics of ρ. M is a delegation policy, which is formally defined in Definition 2. $\varPi$ is a security policy, which is formally defined in Definition 3.

Definition 2 (Delegation policy). Let $\varepsilon(\boldsymbol{c})$ be an any given ground event in $\rho(Q, Q_0, \mathcal{E}, P, M, \varPi)$. M is a formal logic system that specifies whether $\varepsilon_+(\boldsymbol{c})$ or $\varepsilon_-(\boldsymbol{c})$ can be entailed or not in what states and with what system characteristics in ρ. If $\varepsilon_+(\boldsymbol{c})$ (resp. $\varepsilon_-(\boldsymbol{c})$) is entailed by M in the states with the designated characteristics in ρ, the intended delegatees is allowed to authorize $\varepsilon_+(\boldsymbol{c})$ (resp. $\varepsilon_-(\boldsymbol{c})$) to other subjects in the states with the characteristics in ρ.

Definition 3 (Security policy)**.** Let $\varepsilon(\boldsymbol{c})$ be any given ground event in $\rho(Q, Q_0, \mathcal{E}, P, M, \mathit{\Pi})$. $\mathit{\Pi}$ is a formal logic system that specifies whether $\varepsilon_+(\boldsymbol{c})$ or $\varepsilon_-(\boldsymbol{c})$ are entailed or not in what states and with what system characteristics in ρ. If $\varepsilon_+(\boldsymbol{c})$ (resp. $\varepsilon_-(\boldsymbol{c})$) is entailed by $\mathit{\Pi}$ in the states with the designated characteristics in ρ, $\varepsilon(\boldsymbol{c})$ is allowed (resp. denied) in the states with the characteristics in ρ.

In $\rho(Q, Q_0, \mathcal{E}, P, M, \mathit{\Pi})$, $\mathit{\Pi}$ should follow the delegation restrictions defined in M, which is called compliance. Based on Definition 2 and Definition 3, we can easily give its formal definition.

Definition 4 (Compliance)**.** Let $\varepsilon(\boldsymbol{c})$ be any given ground event in $\rho(Q, Q_0, \mathcal{E}, P, M, \mathit{\Pi})$. M and $\mathit{\Pi}$ use the same set of inference meta-rules. Suppose (1) if $\varepsilon_+(\boldsymbol{c})$ is entailed by $\mathit{\Pi}$ in the states with the designated characteristics in ρ, $\varepsilon_+(\boldsymbol{c})$ is also entailed by M in the same states with the same characteristics in ρ, and (2) if $\varepsilon_-(\boldsymbol{c})$ is entailed by M in the states with the designated characteristics in ρ, $\varepsilon_-(\boldsymbol{c})$ is also entailed by $\mathit{\Pi}$ in the same states with the same characteristics in ρ. Then we say $\mathit{\Pi}$ is compliant with M, or M-compliant.

Intuitively, compliance depicts the inherent constraint relationship between M and $\mathit{\Pi}$. Definition 4 ensure the ground event atoms that are allowed by ρ should also be allowed both by M and $\mathit{\Pi}$ and must not be denied by M. That's to say, M absolutely dominates $\mathit{\Pi}$. For first-order logic system, different inference meta-rules may entail different logic results. So we claim that M and $\mathit{\Pi}$ use the same set of inference meta-rules.

3.2 Policy declaration and policy framework

From Definition 3, we claim that a full sense of security policy in an information system should have two fundamental functions: depicting its characteristics and guarding its behaviors based on the characteristics. In this paper, for the sake of simplicity, the characteristics are abstractly depicted through a set of rules, and the behaviors are depicted by the events.

With Definition 2 and Definition 3, we claim that delegation policy is a special kind of security policy. Hence, They can be declared in the same way. There are many proposed policy languages [14–19,21]. However, with the merits of logic program, such as reasoning ability, declarative semantic and simplicity, the logic program based policy languages are popular [14–18]. Hence, in this paper, we suppose security policy and delegation policy are declared by logic programs. In the following subsections, we will introduce the language to declare security policy, the semantic of security policy and the query evaluation algorithm with conflict and unsettlement resolution against security policy. All of these compose a security policy framework.

3.2.1 *Logic program based policy declaration*

In a logic program based policy language, policy is declared by one or several rules as follow

$$A \leftarrow L_1, \ldots, L_n. \tag{F-1}$$

where A in the rule head is any atom, L_i in the rule body is a positive or negative literal, and $n \geqslant 0$. Formula (F-1) is an equivalent variant of the following logic formula

$$\forall(A \leftarrow L_1 \wedge \cdots \wedge L_n) \qquad \text{or} \qquad A \leftarrow L_1 \wedge \cdots \wedge L_n. \tag{F-2}$$

Definition 5 (Security policy)**.** A set of logic formulas of the forms of Formula (F-1).

Obviously, each security policy is a logic program. For simplicity, for any given $\rho(Q, Q_0, \mathcal{E}, P, M, \mathit{\Pi})$, we only focus on the scenario where $\mathit{\Pi}$ in ρ can be partitioned into two disjoint subsets as follow:

1. System control rules base (SCB): the predicates appearing in the heads of the rules in SCB must be event predicates belonging to $\mathcal{E}$, and the predicates in the bodies must be characteristic predicates which belong to $\mathit{\Omega}$. Essentially, the rules in SCB are used to regulate the events in ρ, such as pr_1 and pr_2 in security policy 1 shown in Figure 2, and $pr_1 - pr_3$ in security policy 2 shown in Figure 5.

2. System invariant rules base (SIB): the predicates appearing in the rules in SIB must be characteristic predicates belonging to $\mathit{\Omega}$. Essentially, SIB depicts the invariant properties existing in ρ, such as the

facts in ρ, the relationships in the entities. So the rules in SIB are also called invariants, such as $pr_4 - pr_8$ in security policy 2 shown in Figure 5.

As we have pointed out, delegation policy is a special kind of security policy with particular purpose. For simplicity, we define delegation policy as a set of rules whose head must be event atoms.

Definition 6 (Delegation policy). A set of logic formulas of the form $\varepsilon_+(\boldsymbol{t}) \leftarrow L_1, \ldots, L_n$ or $\varepsilon_-(\boldsymbol{t}) \leftarrow L_1, \ldots, L_n$, where $\varepsilon_+(\boldsymbol{t})$ (resp. $\varepsilon_-(\boldsymbol{t})$) in the rule head is an allowed (resp. denied) event atom.

As we know, the set of rules in M are used to depict the privileges that the intended delegatees can authorize to other subjects, which is also called SCB of M. Comparing Definition 5 and Definition 6, we can see that delegation policy does not have SIB to depict the characteristics of its intended information system. This is because that (1) the essence of delegation policy is to specify the set of ground event atoms that the delegatees can authorize to other subjects, SCB is enough to do so; (2) the delegators usually declare the delegation policies abstractly, but they need not or have no context to designate the characteristics of its intended information system; (3) the delegators usually trust their intended delegatees on the declarations of system characteristics. That's to say, M and $\mathit{\Pi}$ share the same characteristics which will be declared in $\mathit{\Pi}$ by the intended delegatees. Hence, we claim that it is reasonable to suppose that the head of each rule in delegation policy is an allowed or denied event atom.

3.2.2 *Semantics of security policy and delegation policy*

As we have pointed out, a full sense of security policy should have two fundamental functions: depicting its characteristics and guarding its events. By Definition 5, each security policy is a full sense because it has both SCB and SIB. Hence, it is easy to define its formal semantic. We stipulate that the semantic of $\mathit{\Pi}$ is the semantic of its corresponding logic program. For any given logic program, it suffices to only research on Herbrand semantics [20] (page 17, proposition 3.2). However, a logic program may have many types of semantics defined in different scenarios, such as minimal model(of definite logic program), perfect model (of stratified logic program) [22], stable model [23], well-founded model [24] and so on.

Unfortunately, there are still many different opinions on what is the canonical semantic of a logic program or how to definite it [24]. Hence, we don't explicitly designate the semantic of a security policy. Instead, we suppose there exists a complete and sound query evaluation algorithm LpQuery($\mathcal{P}, Q$) which implicitly defines its formal semantic, where $\mathcal{P}$ is a security policy, and Q is a query goal. For example, if we think stable model is canonical, we can designated LpQuery($\mathcal{P}, Q$) as Smodels($\mathcal{P}, Q$), where Smodels [25] is a sound and complete query evaluation algorithm under the stable model semantics.

Let $\mathbb{S}$ be a Herbrand semantic of $\mathcal{P}$. $\mathbb{S}$ is supported if for any ground atom $a \in \mathbb{S}$, there exists a ground rule $a \leftarrow a_1, \ldots, a_n$ instantiated from a rule in $\mathcal{P}$ and all $a_i \in \mathbb{S}$. Most of the Herbrand semantics are supported, such as those mentioned above. LpQuery($\mathcal{P}, Q$) returns a set of all the ground substitutions that make Q a logic result of $\mathcal{P}$. If Q is a ground atom a, then LpQuery($\mathcal{P}, a$) returns true (when a is a logic result of $\mathcal{P}$) or false (when a is not a logic result of $\mathcal{P}$).

However, delegation policy isn't full-fledged because it has no SIB to depict the characteristics of its intended information systems. Hence, its Herbrand semantic could not show its real intention. We claim that the characteristics of an information system are not only the context of its security policy, but also the context of its delegation policy. That's to say, for any given $\rho(Q, Q_0, \mathcal{E}, P, M, \mathit{\Pi})$, the real sense delegation policy is $M^{\mathit{\Pi}} = M \cup \mathrm{SIB}(\mathit{\Pi})$. Obviously, $M^{\mathit{\Pi}}$ is full-fledged. Since $M^{\mathit{\Pi}}$ can be constituted uniquely by M and $\mathit{\Pi}$, we still use M to denote $M^{\mathit{\Pi}}$ without making any confusion.

In the following sections, Let the semantics of $\mathit{\Pi}$ and M be $\mathbb{S}_{\mathit{\Pi}}$ and $\mathbb{S}_M$, and their query evaluation algorithm be LpQuery. Let $\mathit{\Gamma}$ denote $\mathit{\Pi}$ or M, and $\varepsilon(\boldsymbol{c})$ be an ground event atom.

Definition 7 (Conflict). $\mathit{\Gamma}$ is $\varepsilon(\boldsymbol{c})$ conflicted if $\varepsilon_+(\boldsymbol{c}) \in \mathbb{S}_{\mathit{\Gamma}}$ and $\varepsilon_-(\boldsymbol{c}) \in \mathbb{S}_{\mathit{\Gamma}}$. $\mathit{\Gamma}$ is ε conflict-free if there is no $\varepsilon(\boldsymbol{c})$ conflict for any $\boldsymbol{c}$. $\mathit{\Gamma}$ is conflict-free if $\mathit{\Gamma}$ is ε conflict-free for any ε.

Definition 8 (Unsettlement). $\mathit{\Gamma}$ is $\varepsilon(\boldsymbol{c})$ unsettled if neither $\varepsilon_+(\boldsymbol{c}) \in \mathbb{S}_{\mathit{\Gamma}}$ nor $\varepsilon_-(\boldsymbol{c}) \in \mathbb{S}_{\mathit{\Gamma}}$. $\mathit{\Gamma}$ is ε unsettlement-free if there is no $\varepsilon(\boldsymbol{c})$ unsettlement for any $\boldsymbol{c}$. $\mathit{\Gamma}$ is unsettlement-free if $\mathit{\Gamma}$ is ε unsettlement-free for any ε.

Unfortunately, it is difficult in practice to construct a conflict-free or unsettlement-free delegation policy and security policy when both allowed and denied event atoms could appear in them. Usually, they are resolved with some default conflict and unsettlement resolution rules.

Conflict resolution: Generally, conflicts are resolved by two conventional classes of default rules [15]: denial taking precedence (DTP) and permission taking precedence (PTP). In our approach, they are declared as Formula (F-3), and stipulate that $\varepsilon(\boldsymbol{c})$ is allowed if $\varepsilon_{++}(\boldsymbol{c}) \in \mathbb{S}_{\Gamma}$, and $\varepsilon(\boldsymbol{c})$ is denied if $\varepsilon_{--}(\boldsymbol{c}) \in \mathbb{S}_{\Gamma}$.

$$\begin{aligned} &\text{PTP}: \quad \varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X}), \\ &\text{DTP}: \quad \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X}), \qquad \text{(F-3)} \\ &\text{OWA}: \quad \varepsilon_{++}(\boldsymbol{X}) \leftarrow \mathit{not}\ \varepsilon_{-}(\boldsymbol{X}), \\ &\text{CWA}: \quad \varepsilon_{--}(\boldsymbol{X}) \leftarrow \mathit{not}\ \varepsilon_{+}(\boldsymbol{X}). \qquad \text{(F-4)} \end{aligned}$$

Unsettlement resolution: Generally, unsettlements are also resolved by two conventional classes of default rules [15]: close world assumption (CWA) and open world assumption (OWA). CWA means that everything that is true is explicitly stated in the program or can be derived from the program. On the contrary, OWA means that everything that is false is explicitly stated in the program or can be derived from the program. In our approach, they can be declared as Formula (F-4).

Conflict and unsettlement resolution is essential to keeping secure. For simplicity, we claim that, for any given event ε, there is one and only one default conflict resolution rule which is either PTP or DTP and one and only one default unsettlement resolution rule which is either OWA or CWA in M or Π. The set of all default conflict resolution rules in a policy is called conflict resolution base (CRB); and the set of all default unsettlement resolution rules in a policy is called unsettlement resolution base (URB). Note that CRB and URB may be implicit in a policy.

3.2.3 *Query evaluation with conflict and unsettlement resolution*

Before $\rho(Q, Q_0, \mathcal{E}, P, M, \Pi)$ executes a ground event $\varepsilon(\boldsymbol{c})$, it should queries against Π with goal "$\leftarrow \varepsilon_{+}(\boldsymbol{c})$". The query evaluation algorithm is illustrated in Algorithm 1. If the algorithm returns true, $\varepsilon(\boldsymbol{c})$ is allowed, else $\varepsilon(\boldsymbol{c})$ is denied. At the same time, it can decide whether there is $\varepsilon(\boldsymbol{c})$ conflict (if bConflicted is *true*) or unsettlement (if bUnsettled is *true*) in $\text{SCB}(\Pi) \cup \text{SIB}(\Pi)$.

```
Algorithm 1 bool PolicyQuery ($\varepsilon(\boldsymbol{c})$, $\Pi$, bConflicted, bUnsettled).
bConflicted := false; bUnsettled:=false;
let $b_1$ =LpQuery(SCB($\Pi$) $\cup$ SIB($\Pi$), $\varepsilon_{+}(\boldsymbol{c})$);
let $b_2$ =LpQuery(SCB($\Pi$) $\cup$ SIB($\Pi$), $\varepsilon_{-}(\boldsymbol{c})$);
if $b_1$ =true and $b_2$=true then
    bConflicted:=true, and
    select out the one and the only one conflict resolution rule $cr$ from CRB($\Pi$),
      which is $\varepsilon_{++}(\boldsymbol{x}) \leftarrow \varepsilon_{+}(\boldsymbol{x}), \varepsilon_{-}(\boldsymbol{x})$ or $\varepsilon_{--}(\boldsymbol{x}) \leftarrow \varepsilon_{+}(\boldsymbol{x}), \varepsilon_{-}(\boldsymbol{x})$;
    if $cr$ is $\varepsilon_{++}(\boldsymbol{x}) \leftarrow \varepsilon_{+}(\boldsymbol{x}), \varepsilon_{-}(\boldsymbol{x})$, then PolicyQuery returns true, else PolicyQuery returns false;
end if
if $b_1$ = false and $b_2$ =false then
    bUnsettled := true, and
    select out the one and the only one unsettlement resolution rule $ur$ from URB($\Pi$),
      which is $\varepsilon_{++}(\boldsymbol{x}) \leftarrow not\ \varepsilon_{-}(\boldsymbol{x})$ or $\varepsilon_{--}(\boldsymbol{x}) \leftarrow not\ \varepsilon_{+}(\boldsymbol{x})$.
    if $ur$ is $\varepsilon_{++}(\boldsymbol{x}) \leftarrow not\ \varepsilon_{-}(\boldsymbol{x})$, then PolicyQuery returns true, else PolicyQuery returns false.
end if
if $b_1$ =true and $b_2$ =false, then PolicyQuery returns true;
if $b_1$ =false and $b_2$ =true, then PolicyQuery returns false
```

Obviously, PolicyQuery always terminates and returns true or false if the underlying logic program query evaluation algorithm LpQuery always terminates. Suppose the computational complexity of Lp-

Query is δ. Then the computational complexity of Algorithm 1 is 2δ+sizeof(CRB(Π))+sizeof(URB(Π))) $\approx 2\delta$.

Theorem 1. For any given ground event atom $\varepsilon(\boldsymbol{c})$, suppose the algorithm LpQuery in Algorithm 1 is sound, complete and terminated. Then the security policy Π, which guards ε, will allow or deny $\varepsilon(\boldsymbol{c})$ explicitly. That's to say, Algorithm 1 is conflict resolved and unsettlement resolved.

Proof. Straight forward.

Let $\Pi_r = \Pi \cup \{\varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{-}(\boldsymbol{X}); \qquad \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{-}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{+}(\boldsymbol{X}).\ |\varepsilon \in \mathcal{E}\}$. Then we have

Theorem 2. Let $\mathbb{S}_{\Pi_r}$ be the semantic of Π_r. For any ground event atom $\varepsilon(\boldsymbol{c})$, either $\varepsilon_{++}(\boldsymbol{c})$ or $\varepsilon_{--}(\boldsymbol{c})$ does come into existence in $\mathbb{S}_{\Pi_r}$, but not both.

Proof. The union of the definitions of ε_{++} and ε_{--} in Π_r must be one of the following four sets.

$$\left\{\begin{array}{l} \varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{-}(\boldsymbol{X});\\ \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{-}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{+}(\boldsymbol{X});\\ \varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X});\\ \varepsilon_{++}(\boldsymbol{X}) \leftarrow \mathit{not}\ \varepsilon_{-}(\boldsymbol{X}). \end{array}\right\}, \quad \left\{\begin{array}{l} \varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{-}(\boldsymbol{X});\\ \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{-}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{+}(\boldsymbol{X});\\ \varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X});\\ \varepsilon_{--}(\boldsymbol{X}) \leftarrow \mathit{not}\ \varepsilon_{+}(\boldsymbol{X}). \end{array}\right\},$$

$$\left\{\begin{array}{l} \varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{-}(\boldsymbol{X});\\ \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{-}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{+}(\boldsymbol{X});\\ \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X});\\ \varepsilon_{++}(\boldsymbol{X}) \leftarrow \mathit{not}\ \varepsilon_{-}(\boldsymbol{X}). \end{array}\right\}, \quad \left\{\begin{array}{l} \varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{-}(\boldsymbol{X});\\ \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{-}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{+}(\boldsymbol{X});\\ \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X});\\ \varepsilon_{--}(\boldsymbol{X}) \leftarrow \mathit{not}\ \varepsilon_{+}(\boldsymbol{X}). \end{array}\right\}.$$

For any ground event atom $\varepsilon(\boldsymbol{c})$, there is one and only one ground rule instantiated from one of the above sets, whose truth value is true. That's to say, $\varepsilon_{++}(\boldsymbol{c})$ or $\varepsilon_{--}(\boldsymbol{c})$ must be in $\mathbb{S}_{\Pi'}$, but not both.

Theorem 3. Let $b = \text{PolicyQuery}(\Pi, \varepsilon(\boldsymbol{c}), -, -)$, $b' = \text{LpQuery}(\Pi_r, \varepsilon_{++}(\boldsymbol{c}))$. Then $b \equiv b'$.

Proof. We prove the theorem in following two cases:

(1) Suppose b =true, according to Algorithm 1.

a) if b_1 =true and b_2 =true, then there is $\varepsilon(\boldsymbol{c})$ conflict in Π and the one and the only one resolution rule must be PTP: $\varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X})$. Hence, the definition of ε_{++} in Π_r includes $\varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X})$. Since b_1 =true and b_2 =true, i.e., $\text{SCB}(\Pi) \cup \text{SIB}(\Pi) \models \varepsilon_{+}(\boldsymbol{c})$ and $\text{SCB}(\Pi) \cup \text{SIB}(\Pi) \models \varepsilon_{-}(\boldsymbol{c})$, we can easily get $\Pi_r \models \varepsilon_{+}(\boldsymbol{c})$ and $\Pi_r \models \varepsilon_{-}(\boldsymbol{c})$. Hence, $\Pi' \models \varepsilon_{++}(\boldsymbol{c})$, i.e., b' =true;

b) if b_1 =false and b_2 =false, then $\varepsilon(\boldsymbol{c})$ is unsettled in Π and the one and the only one resolution rule must be OWA: $\varepsilon_{++}(\boldsymbol{X}) \leftarrow \mathit{not}\varepsilon_{-}(\boldsymbol{X})$. Hence, the definition of ε_{++} in Π_r includes $\varepsilon_{++}(\boldsymbol{X}) \leftarrow \mathit{not}\varepsilon_{-}(\boldsymbol{X})$. Since b_1 =false and b_2 =false, i.e., $\text{SCB}(\Pi) \cup \text{SIB}(\Pi) \not\models \varepsilon_{+}(\boldsymbol{c})$ and $\text{SCB}(\Pi) \cup \text{SIB}(\Pi) \not\models \varepsilon_{-}(\boldsymbol{c})$, we can easily get $\Pi_r \models \varepsilon_{+}(\boldsymbol{c})$. Hence, $\Pi_r \models \varepsilon_{++}(\boldsymbol{c})$, i.e., b' =true;

c) if b_1 =true and b_2 =false, then there is no $\varepsilon(\boldsymbol{c})$-conflict and no $\varepsilon(\boldsymbol{c})$-unsettlement. However, the definition of ε_{++} in Π_r includes $\varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \mathit{not}\ \varepsilon_{-}(\boldsymbol{X})$. Since b_1 =true and b_2 =false, i.e., $\text{SCB}(\Pi) \cup \text{SIB}(\Pi) \models \varepsilon_{+}(\boldsymbol{c})$ and $\text{SCB}(\Pi) \cup \text{SIB}(\Pi) \not\models \varepsilon_{-}(\boldsymbol{c})$, we can easily get $\Pi_r \models \varepsilon_{+}(\boldsymbol{c})$ and $\Pi_r \not\models \varepsilon_{-}(\boldsymbol{c})$. Hence, $\Pi_r \models \varepsilon_{++}(\boldsymbol{c})$, i.e., b' =true;

d) since b=true, it is impossible that b_1 =false and b_2 =true. Hence b' =true.

(2) Suppose b =false, according to Algorithm 1, if b_1=true and b_2 =true, then there is $\varepsilon(\boldsymbol{c})$-conflict in Π and the one and the only one resolution rule is PTP : $\varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X})$. Hence, the definition of ε_{--} in Π_r includes $\varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_{+}(\boldsymbol{X}), \varepsilon_{-}(\boldsymbol{X})$. Since b_1 =true and b_2 =true, i.e., $\text{SCB}(\Pi) \cup \text{SIB}(\Pi) \models \varepsilon_{+}(\boldsymbol{c})$ and $\text{SCB}(\Pi) \cup \text{SIB}(\Pi) \models \varepsilon_{-}(\boldsymbol{c})$, we can easily get $\Pi_r \models \varepsilon_{+}(\boldsymbol{c})$ and $\Pi_r \models \varepsilon_{-}(\boldsymbol{c})$. Hence, $\Pi_r \models \varepsilon_{--}(\boldsymbol{c})$. According Theorem 2, $\Pi_r \models \varepsilon_{++}(\boldsymbol{c})$ i.e., b' = false. In the other cases, the proving procedures are similar, we omit them here.

Vice versa, the proving procedure is similar, due to space limitations, we omit it here.

With Theorem 3, we can know that PolicyQuery is logically equivalent to LpQuery if we don't care about the conflict and unsettlement detection. Ultimately, we construct the logic framework for security policy declaration, query evaluation, and conflict and unsettlement resolution.

4 Constructing compliant security polity

Firstly, we propose a logic program equivalent transformation algorithm. Based on this algorithm, we discuss the security policy rewriting algorithm in accordance with the principle from special to general.

4.1 Logic program equivalent transformation

Let $\mathrm{Def}_P(p)$ be the set of all k rules in logic program P with predicate symbol p in their heads, called the definition of p in P. Let $p(t_1,\ldots,t_n) \leftarrow l_1,\ldots,l_m$ be any rule in $\mathrm{Def}_P(p)$. We require a new predicate symbol $=$, not appearing in P, whose intended interpretation is the identity relation under equality theory [20] (page 78, 79). We transform every rule in $\mathrm{Def}_P(p)$ into the form

$$p(X_1,\ldots,X_n) \leftarrow \exists_{\boldsymbol{Y}}((X_1=t_1)\wedge\cdots\wedge(X_n=t_n)\wedge l_1\wedge\cdots\wedge l_k),$$

where $X_1,\ldots,X_n$ are the variables not appearing in the original rule, $\boldsymbol{Y}$ is the sequence of all the free variables in the original rule. Then we obtain a formula of the form

$$\begin{aligned} p(X_1,\ldots,X_n) \leftarrow\ & \exists_{\boldsymbol{Y}_1}(X_1=t_{1,1}\wedge\cdots\wedge X_n=t_{1,\,n}\wedge l_{1,\,1}\wedge\cdots\wedge l_{1,k_1}) \\ & \vee\cdots \\ & \vee\exists_{\boldsymbol{Y}_k}(X_1=t_{k,1}\wedge\cdots\wedge X_n=t_{k,n}\wedge l_{k,1}\wedge\cdots\wedge l_{k,k_k}). \end{aligned} \tag{F-5}$$

This is denoted by $\mathrm{Def}_P^{\mathcal{F}}(p)$. With Theorem 4, we also call $\mathrm{Def}_P^{\mathcal{F}}(p)$ the definition of p in P.

Theorem 4. $\mathbb{S}$ is a supported Herbrand semantic of P if and only if $\mathbb{S}$ is a supported Herbrand semantic of $(P-\mathrm{Def}_P(p))\cup\{\mathrm{Def}_P^{\mathcal{F}}(p)\}$. That's to say, P is logically equivalent to $(P-\mathrm{Def}_P(p))\cup\{\mathrm{Def}_P^{\mathcal{F}}(p)\}$ under any supported Herbrand semantic.

Proof. (Reductio ad absurdum)

($\Rightarrow$): Suppose $\mathbb{S}$ is not a supported Herbrand semantic of $(P-\mathrm{Def}_P(p))\cup\{\mathrm{Def}_P^{\mathcal{F}}(p)\}$. Then there is at least one formula $p(\boldsymbol{c})\leftarrow F$ which is an instantiation of $\mathrm{Def}_P^{\mathcal{F}}(p)$, such that $p(\boldsymbol{c})\notin\mathbb{S}$, but there is a conjunctive sub-formula $l_1\wedge\cdots\wedge l_k$ of F with truth value true under $\mathbb{S}$. Hence, the truth of formula $p(\boldsymbol{c})\leftarrow l_1\wedge\cdots\wedge l_k$ is false in $\mathbb{S}$. However, the formula is an instantiation of some rule in P. That's to say, $\mathbb{S}$ is not a supported Herbrand semantic of P. Contradiction.

($\Leftarrow$): Suppose $\mathbb{S}$ is not a supported Herbrand semantic of P. Then there is at least one ground rule $p(\boldsymbol{c})\leftarrow l_1\wedge\cdots\wedge l_k$ which is an instantiation of some rule in P, such that $p(\boldsymbol{c})\notin\mathbb{S}$, but the truth of $l_1\wedge\cdots\wedge l_k$ is true. Since $p(\boldsymbol{c})\notin\mathbb{S}$, the truth of all the ground rules instantiated from $\mathrm{Def}_P(p)$ with $p(\boldsymbol{c})$ as their heads is false. Hence, the truth of $p(X_1,\ldots,X_n)\leftarrow\exists_{\boldsymbol{Y}_1}(\boldsymbol{c}=\boldsymbol{t}_1\wedge l_{1,\,1}\wedge\cdots\wedge l_{1,\,k_1})\vee\cdots\vee\exists_{\boldsymbol{Y}_k}(\boldsymbol{c}=\boldsymbol{t}_k\wedge l_{k,1}\wedge\cdots\wedge l_{k,k_k})$ is false, which is instantiated from $\mathrm{Def}_P^{\mathcal{F}}(p)$. That's to say, $\mathbb{S}$ is not a supported Herbrand semantic of $(P-\mathrm{Def}_P(p))\cup\{\mathrm{Def}_P^{\mathcal{F}}(p)\}$. It is a contradiction.

4.2 Rewriting algorithm

Let $\rho(Q,Q_0,\mathcal{E},P,M,\Pi)$ be an information system. In this section, we will propose the security policy rewriting algorithm which automatically rewrite Π into a compliant security policy Π_M with M. We will introduce the rewriting algorithm in five cases.

Case 1: For any given event ε, both Π and M have no rules with event atom $\varepsilon_-(\boldsymbol{t})$ but some rules with event atom $\varepsilon_+(\boldsymbol{t})$ appearing in their heads. In this case, there is one and only one implicit ε-unsettlement resolution rule $\varepsilon_-(\boldsymbol{X})\leftarrow \mathit{not}\ \varepsilon_+(\boldsymbol{X})$ in both Π and M. Obviously, there are no ε conflicts in Π and M. Hence, Π and M do not need ε conflict resolution rule. With Theorem 4, the definitions of ε_+ and ε_- in Π can be denoted by $\mathrm{Def}_\Pi^{\mathcal{F}}(\varepsilon_+)=\varepsilon_+(\boldsymbol{X})\leftarrow F_\Pi^+$ and $\mathrm{Def}_\Pi^{\mathcal{F}}(\varepsilon_-)=\varepsilon_-(\boldsymbol{X})\leftarrow \mathit{not}\ \varepsilon_+(\boldsymbol{X})$, and the definitions of ε_+ and ε_- in M can be denoted by $\mathrm{Def}_M^{\mathcal{F}}(\varepsilon_+)=\varepsilon_+(\boldsymbol{X})\leftarrow F_M^+$ and $\mathrm{Def}_M^{\mathcal{F}}(\varepsilon_-)=\varepsilon_-(\boldsymbol{X})\leftarrow \mathit{not}\ \varepsilon_+(\boldsymbol{X})$, where F_Π^+ and F_M^+ are formulas of the form shown on the right-hand side of formula (F-5).

Based on those, let $\mathrm{Def}_{\Pi_M}^{\mathcal{F}}(\varepsilon_+)=\varepsilon_+(\boldsymbol{X})\leftarrow F_\Pi^+\wedge F_M^+$ and $\mathrm{Def}_{\Pi_M}^{\mathcal{F}}(\varepsilon_-)=\varepsilon_-(\boldsymbol{X})\leftarrow \mathit{not}\ \varepsilon_+(\boldsymbol{X})\vee \mathit{not}\ \varepsilon_+(\boldsymbol{X})$, i.e., $\mathrm{Def}_{\Pi_M}^{\mathcal{F}}(\varepsilon_-)=\varepsilon_-(\boldsymbol{X})\leftarrow \mathit{not}\ \varepsilon_+(\boldsymbol{X})$. Then we construct the definitions of ε_+ and ε_- in Π_M. Obviously, Π_M is ε conflict and unsettlement-free under all the supported Herbrand semantics.

Case 2: For any given event ε, both Π and M have no rules with event atom $\varepsilon_+(\boldsymbol{t})$ but some rules with event atom $\varepsilon_-(\boldsymbol{t})$ appeared in their heads. This case is similar to Case 1. We skip it.

Case 3: For any given event ε, Π has no rules with event atom $\varepsilon_-(\boldsymbol{t})$ but some rules with event atom $\varepsilon_+(\boldsymbol{t})$ appearing in their heads, and M has no rules with event atom $\varepsilon_+(\boldsymbol{t})$ but some rules with event atom $\varepsilon_-(\boldsymbol{t})$ appearing in their heads. In this case, there is one and only one implicit ε-unsettlement resolution rule $\varepsilon_-(\boldsymbol{X}) \leftarrow \textit{not}\ \varepsilon_+(\boldsymbol{X})$ in Π and $\varepsilon_+(\boldsymbol{X}) \leftarrow \textit{not}\ \varepsilon_-(\boldsymbol{X})$ in M. Obviously, there are no ε-conflicts in Π and M. With Theorem 4, the definitions of ε_+ and ε_- in Π can be denoted by $\mathrm{Def}^{\mathcal{F}}_{\Pi}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_{\Pi}$ and $\mathrm{Def}^{\mathcal{F}}_{\Pi}(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow \textit{not}\ \varepsilon_+(\boldsymbol{X})$, and the definitions of ε_+ and ε_- in M can be denoted by $\mathrm{Def}^{\mathcal{F}}_{M}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow \textit{not}\ \varepsilon_-(\boldsymbol{X})$ and $\mathrm{Def}^{\mathcal{F}}_{M}(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_M$, where F^+_{Π}, F^+_M, F^-_{Π} and F^-_M are formulas of the form shown on the right-hand side of formula (F-5).

Based on those, let $\mathrm{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_{\Pi} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{X})$ and $\mathrm{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_M \vee \textit{not}\ \varepsilon_+(\boldsymbol{X})$. Then we construct the definitions of ε_+ and ε_- in Π_M. In this case, Π is ε-conflict free and ε-unsettlement free (after unsettlement resolution). We replace the definitions of ε_+ and ε_- in Π with $\mathrm{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_+)$ and $\mathrm{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_-)$, which transform Π into Π'. Then we have following results.

Theorem 5. In case 3, Π' is ε-conflict/unsettlement free under all the supported Herbrand semantics.

Proof. (Reduction ad absurdum) Let $\mathbb{S}$ be a supported Herbrand semantic of Π'.

Suppose there are ε-conflicts in Π', whose definition should be

$$\mathrm{Def}^{\mathcal{F}}_{\Pi'}(\varepsilon_{\mp}) = \varepsilon_{\mp}(\boldsymbol{X}) \leftarrow (F^+_{\Pi} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{X})) \wedge (F^-_M \textit{not}\ \varepsilon_+(\boldsymbol{X})).$$

Let $\varepsilon_{\mp}(\boldsymbol{c}) \leftarrow (F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c})) \wedge (F^-_M|_{\boldsymbol{c}} \textit{not}\ \varepsilon_+(\boldsymbol{c}))$ be any instantiated formula from $\mathrm{Def}^{\mathcal{F}}_{\Pi'}(\varepsilon_{\mp})$. Supposing $(F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c})) \wedge (F^-_M|_{\boldsymbol{c}} \vee \textit{not}\ \varepsilon_+(\boldsymbol{c}))$ is true in $\mathbb{S}$, $F^+_{\Pi}|_{\boldsymbol{c}}$, $\textit{not}\ \varepsilon_-(\boldsymbol{c})$ and $(F^-_M|_{\boldsymbol{c}} \vee \textit{not}\ \varepsilon_+(\boldsymbol{c}))$ must be true in $\mathbb{S}$. Since $\textit{not}\ \varepsilon_-(\boldsymbol{c})$ is true in $\mathbb{S}$, we can get $\varepsilon_-(\boldsymbol{c}) \notin \mathbb{S}$. Hence, $(F^-_M|_{\boldsymbol{c}} \vee \textit{not}\ \varepsilon_+(\boldsymbol{c}))$ is false in $\mathbb{S}$ because formula $\varepsilon_-(\boldsymbol{c}) \leftarrow F^-_M|_{\boldsymbol{c}} \vee \textit{not}\ \varepsilon_+(\boldsymbol{c})$, which is instantiated from $\mathrm{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_-)$, is true in $\mathbb{S}$. We get a contradiction. Hence, $(F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c})) \wedge (F^-_M|_{\boldsymbol{c}} \vee \textit{not}\ \varepsilon_+(\boldsymbol{c}))$ is false in $\mathbb{S}$. So there is no ε-conflict in Π'.

Suppose there are ε-unsettlements in Π', whose definition is

$$\mathrm{Def}^{\mathcal{F}}_{\Pi'}(\varepsilon_?) = \varepsilon_?(\boldsymbol{X}) \leftarrow \neg((F^+_{\Pi} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{X})) \vee (F^-_M \vee \textit{not}\ \varepsilon_+(\boldsymbol{X}))).$$

Let $\varepsilon_?(\boldsymbol{c}) \leftarrow \neg((F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c})) \vee (F^-_M|_{\boldsymbol{c}} \vee \textit{not}\ \varepsilon_+(\boldsymbol{c})))$ be any instantiated formula from $\mathrm{Def}^{\mathcal{F}}_{\Pi'}(\varepsilon_?)$. Suppose $(F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c})) \vee (F^-_M|_{\boldsymbol{c}} \vee \textit{not}\ \varepsilon_+(\boldsymbol{c}))$ be false in $\mathbb{S}$. Then $(F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c}))$, $F^-_M|_{\boldsymbol{c}}$ and $\textit{not}\ \varepsilon_+(\boldsymbol{c})$ must be false in $\mathbb{S}$. However, Since $\textit{not}\ \varepsilon_+(\boldsymbol{c})$ is false in $\mathbb{S}$, we can get $\varepsilon_+(\boldsymbol{c}) \in \mathbb{S}$. Hence $(F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c}))$ is true in $\mathbb{S}$ because formula $\varepsilon_+(\boldsymbol{c}) \leftarrow F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c})$, which is instantiated from $\mathrm{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_+)$, is true in $\mathbb{S}$. We get a contradiction. Hence, $(F^+_{\Pi}|_{\boldsymbol{c}} \wedge \textit{not}\ \varepsilon_-(\boldsymbol{c})) \wedge (F^-_M|_{\boldsymbol{c}} \vee \textit{not}\ \varepsilon_+(\boldsymbol{c}))$ is true in $\mathbb{S}$. So there is no ε-unsettlement in Π'.

Case 4: Π has no rules with event atom $\varepsilon_+(\boldsymbol{t})$ but some rules with event atom $\varepsilon_-(\boldsymbol{t})$ appearing in their heads, and M has no rules with event atom $\varepsilon_-(\boldsymbol{t})$ but some rules with event atom $\varepsilon_+(\boldsymbol{t})$ appearing in their heads. This case is similar to Case 3. We skip it.

Case 5: Both Π and M have some rules with event atom $\varepsilon_+(\boldsymbol{t})$ and some other rules with event atom $\varepsilon_-(\boldsymbol{t})$ appearing in their heads. Obviously, Case 1 to Case 4 are special kind of this case. In this case, obviously, it is possible that there exist ε-conflicts or ε-unsettlements in Π and M, which must be resolved. Let Π and M be conflict and unsettlement resolved with CRB and URB under PolicyQuery algorithm illustrated in Algorithm 1. Hence, there is one and only one default ε-conflict resolution rule which must be $\varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_+(\boldsymbol{X}), \varepsilon_-(\boldsymbol{X})$ or $\varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_+(\boldsymbol{X}), \varepsilon_-(\boldsymbol{X})$, and one and only one default ε-unsettlement resolution rule which must be $\varepsilon_{++}(\boldsymbol{X}) \leftarrow \textit{not}\ \varepsilon_-(\boldsymbol{X})$ or $\varepsilon_{--}(\boldsymbol{X}) \leftarrow \textit{not}\ \varepsilon_+(\boldsymbol{X})$. Note that these rules do explicitly bind to Π and M, and Π and M have the same CRB and URB.

With Theorem 4, suppose the definitions of ε_+ and ε_- in Π can be denoted by $\mathrm{Def}^{\mathcal{F}}_{\Pi}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_{\Pi}$ and $\mathrm{Def}^{\mathcal{F}}_{\Pi}(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_{\Pi}$, and the definitions of ε_+ and ε_- in M can be denoted by $\mathrm{Def}^{\mathcal{F}}_{M}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_M$ and $\mathrm{Def}^{\mathcal{F}}_{M}(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_M$. Let $\mathrm{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_M \wedge F^+_{\Pi}$ and $\mathrm{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_M \vee F^-_{\Pi}$. Then we construct the definitions of ε_+ and ε_- in Π_M.

The definition of ε-conflict in $\mathit{\Pi}_M$ should be $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}_M}(\varepsilon_{\mp}) = \varepsilon_{\mp}(\boldsymbol{X}) \leftarrow (F^+_M \wedge F^+_{\mathit{\Pi}}) \wedge (F^-_M \vee F^-_{\mathit{\Pi}})$ which is equivalent to $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}_M}(\varepsilon_{\mp}) = \varepsilon_{\mp}(\boldsymbol{X}) \leftarrow (F^+_M \wedge F^-_M \wedge F^+_{\mathit{\Pi}}) \vee (F^+_M \wedge F^+_{\mathit{\Pi}} \wedge F^-_{\mathit{\Pi}})$. If there is no ε conflict in $\mathit{\Pi}$ and M, the truths of $F^+_M \wedge F^-_M$ and $F^+_{\mathit{\Pi}} \wedge F^-_{\mathit{\Pi}}$ are always false. Hence, $\mathit{\Pi}_M$ is ε conflict-free. Similarly, the definition of ε-unsettlement in $\mathit{\Pi}_M$ should be $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}_M}(\varepsilon_{?}) = \varepsilon_{?}(\boldsymbol{X}) \leftarrow \neg((F^+_M \wedge F^+_{\mathit{\Pi}}) \vee (F^-_M \vee F^-_{\mathit{\Pi}}))$ which is equivalent to $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}_M}(\varepsilon_{?}) = \varepsilon_{?}(\boldsymbol{X}) \leftarrow \neg((F^-_M \vee F^+_{\mathit{\Pi}} \vee F^-_{\mathit{\Pi}}) \wedge (F^+_M \vee F^-_M \vee F^-_{\mathit{\Pi}}))$. If there is no ε unsettlement in $\mathit{\Pi}$ and M, the truths of $F^+_M \vee F^-_M$ and $F^+_{\mathit{\Pi}} \vee F^-_{\mathit{\Pi}}$ are always true. Hence, $\mathit{\Pi}_M$ is ε unsettlement-free. However, if there is ε conflicts/unsettlements in $\mathit{\Pi}$ or M, there could be ε conflicts/unsettlements in $\mathit{\Pi}_M$. This is why we require that $\mathit{\Pi}$ and M are conflict and unsettlement resolved.

To ensure that the final output is correct, our approach requires that:

(1) $\mathit{\Pi}$ and M embed the same CRB and URB; and

(2) for every event ε, there is one and only one default PTP or DTP conflict resolution rule in CRB and one and only one default CWA or OWA unsettlement resolution rule in URB; and

(3) the predicates appearing in the bodies of the rules in SCB($\mathit{\Pi}$) and SCB(M) must also appear in SIB($\mathit{\Pi}$); the predicates appearing in SIB($\mathit{\Pi}$) may not appear in SCB($\mathit{\Pi}$) or SCB(M).

Definition 9 (Noninterference)**.** If a policy satisfies above three conditions, it is noninterference.

For any given $\varepsilon \in \mathcal{E}$ we can get $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}_M}(\varepsilon_+)$ and $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}_M}(\varepsilon_+)$ according to its definitions in $\mathit{\Pi}$ and M with above approach. Then, we transform $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}_M}(\varepsilon_+)$ and $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}_M}(\varepsilon_-)$ with the algorithm introduced in Chapter 4 of [20] into two set of logic program rules.

After all $\mathrm{Def}_{\mathit{\Pi}}(\varepsilon_+)$ and $\mathrm{Def}_{\mathit{\Pi}}(\varepsilon_-)$ in $\mathit{\Pi}$ are transformed into sets of logic program rules, $\mathit{\Pi}$ is transformed into a new security policy $\mathit{\Pi}_M$. With Theorem 3, we can know that PolicyQuery($\mathit{\Pi}_M, \varepsilon(\boldsymbol{c}), -, -$) is logically equivalent to LpQuery($\mathit{\Pi}_{M_r}, \varepsilon_{++}(\boldsymbol{c})$) if LpQuery is sound and complete, where $\mathit{\Pi}_{M_r} = \mathit{\Pi}_M \cup \{\varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_+(\boldsymbol{X}), \mathit{not}\ \varepsilon_-(\boldsymbol{X}). \varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_-(\boldsymbol{X}), \mathit{not}\ \varepsilon_+(\boldsymbol{X}). | \varepsilon \in \mathcal{E}\}$. With Theorem 1, we can know that $\mathit{\Pi}_{M_r}$ is conflict and unsettlement resolved if its semantic decided by LpQuery is supported.

According to above discussions, we propose the following Algorithm 2. Given conflicts and unsettlements resolved security policy $\mathit{\Pi}$ and delegation policy M, Algorithm 2 can generate a security policy $\mathit{\Pi}_M$ which is compliant with M and maximally reserves $\mathit{\Pi}$'s valid semantic. Obviously, since the algorithm introduced in Chapter 4 of [20] is terminal, Algorithm 2 is also terminal. From Algorithm 2 we can know that M, $\mathit{\Pi}$ and $\mathit{\Pi}_M$ have the same SIB, CRB and URB, but they have different SCB. That's to say, a delegator and its intended delegatees share the same facts, the principle of conflict and unsettlement resolution, but they may have differently actual requirement on how to control a real information system.

Algorithm 2 PolicyRewrite ([in] M, [in] $\mathit{\Pi}$, [out] $\mathit{\Pi}_M$).

```
$P := \emptyset$;    $\mathit{\Pi}_M := \emptyset$;
if $M$ or $\mathit{\Pi}$ are not noninterference, then return.
for each event predicate $\varepsilon_+$ and $\varepsilon_-$ defined in $M$ or $\mathit{\Pi}$:
    with the approach introduced in subsection 4.1, we can get
            $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_{\mathit{\Pi}}$;    $\mathrm{Def}^{\mathcal{F}}_{\mathit{\Pi}}(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_{\mathit{\Pi}}$;
            $\mathrm{Def}_{M^{\mathcal{F}}}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_M$;    $\mathrm{Def}_{M^{\mathcal{F}}}(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_M$.
    add formula $\varepsilon_+(\boldsymbol{X}) \leftarrow F^+_M \wedge F^+_{\mathit{\Pi}}$ and $\varepsilon_-(\boldsymbol{X}) \leftarrow F^-_M \vee F^-_{\mathit{\Pi}}$ to $P$;
    add formula $\varepsilon_{++}(\boldsymbol{X}) \leftarrow \varepsilon_+(\boldsymbol{X}), \mathit{not}\ \varepsilon_-(\boldsymbol{X})$ and $\varepsilon_{--}(\boldsymbol{X}) \leftarrow \varepsilon_-(\boldsymbol{X}), \mathit{not}\ \varepsilon_+(\boldsymbol{X})$ to $P$;
    remove all rules with $\varepsilon_+$ or $\varepsilon_-$ in their heads from $\mathit{\Pi}$;
end for
transform $P$ into a logic program $\mathit{\Pi}_M$ (Chapter 4 of ref. [20]).
copy $\mathit{\Pi}$ into $\mathit{\Pi}_M$.
```

Lemma 1. For any given ground event atom $\varepsilon(\boldsymbol{c})$, let $\mathbb{S}$ be a supported Herbrand semantics of $\mathit{\Pi}_M$. Then either $\varepsilon_{++}(\boldsymbol{c}) \in \mathbb{S}$ or $\varepsilon_{--}(\boldsymbol{c}) \in \mathbb{S}$, but not both.

Proof. straight forward from Theorem 2.

Theorem 6. Given security policy $\mathit{\Pi}$ and its intended delegation policy M declared in our framework, if they are noninterference, the security policy $\mathit{\Pi}_M$ generated by Algorithm 2 is M-compliant, conflict resolved and unsettlement resolved under LpQuery.

Proof. With Lemma 1, we can get that Π_M is conflict resolved and unsettlement resolved under LpQuery.

Suppose $b = \text{LpQuery}(\Pi_M, \varepsilon_+(\boldsymbol{c}))$ and $\text{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_M \wedge F^+_\Pi$. If b =true, we can get $F^+_M|_{\boldsymbol{X}=\boldsymbol{c}}$ and $F^+_\Pi|_{\boldsymbol{X}=\boldsymbol{c}}$ are both true under the semantic of Π_M decided by LpQuery. Since the predicates appeared in F^+_M are all appear in $\text{SIB}(\Pi)$, $F^+_M|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under the semantic of $\text{SIB}(\Pi)$ decided by LpQuery. Hence, $F^+_M|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under the semantic $\mathbb{S}_M$ of M decided by LpQuery. Since $\mathbb{S}_M$ is supported and $[\varepsilon_+(\boldsymbol{X}) \leftarrow F^+_M] \in M$, we can conclude that $\varepsilon_+(\boldsymbol{c}) \in \mathbb{S}_M$, that's to say, true $= \text{LpQuery}(M, \varepsilon_+(\boldsymbol{c}))$.

Suppose $b = \text{LpQuery}(M, \varepsilon_-(\boldsymbol{c}))$ and $\text{Def}^{\mathcal{F}}_M(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_M$. If b =true, $F^-_M|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under the semantic of M decided by LpQuery. Since the predicates appearing in F^-_M all appear in $\text{SIB}(\Pi)$, $F^-_M|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under the semantic of $\text{SIB}(\Pi)$ decided by LpQuery. With M and Π_M sharing $\text{SIB}(\Pi)$, we can conclude that $F^-_M|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under the semantic $\mathbb{S}_{\Pi_M}$ of Π_M decided by LpQuery.

With Definition 4, we can say that Π_M is M-compliant.

Note that, if Π or M is not noninterference, Theorem 6 may not be correct.

4.3 Soundness and completeness

If we only care about compliance, we would be led into a blind alley. For example, the empty security policy $\emptyset$ is compliant with any delegation policy, but it is senseless. This section we will prove $\text{PolicyRewrite}(\Pi, M, \Pi_M)$ not only resolves all incompliance in Π, but also reserves all the valid semantic of Π with M. The former is called soundness, and the latter is called completeness.

Let $\mathbb{S}_\Pi$, $\mathbb{S}_M$ and $\mathbb{S}_{\Pi_M}$ be the semantics of Π, M, and Π_M defined by LpQuery, and they are all supported. Note that, M is an alias of the actual delegation policy $M \cup \text{SIB}(\Pi)$ under the context of Π, so $\mathbb{S}_M$ is an alias of the semantic of $M \cup \text{SIB}(\Pi)$. Then, Soundness and completeness are formally defined as follows.

Definition 10 (Soundness)**.** For any give ground event atom $\varepsilon(\boldsymbol{c})$,

(1) if $\text{LpQuery}(\Pi_M, \varepsilon_+(\boldsymbol{c}))$ =true, then $\text{LpQuery}(\Pi, \varepsilon_+(\boldsymbol{c}))$ =true and $\text{LpQuery}(M, \varepsilon_+(\boldsymbol{c}))$ =true, and

(2) if $\text{LpQuery}(\Pi_M, \varepsilon_-(\boldsymbol{c}))$ =true, then $\text{LpQuery}(\Pi, \varepsilon_-(\boldsymbol{c}))$ =true or $\text{LpQuery}(M, \varepsilon_-(\boldsymbol{c}))$ =true. Then $\text{PolicyRewrite}(\Pi, M, \Pi_M)$ is sound.

Definition 11 (Completeness)**.** For any give ground event atom $\varepsilon(\boldsymbol{c})$,

(1) if $\text{LpQuery}(\Pi, \varepsilon_+(\boldsymbol{c}))$= true and $\text{LpQuery}(M, \varepsilon_+(\boldsymbol{c}))$ =true, then $\text{LpQuery}(\Pi_M, \varepsilon_+(\boldsymbol{c}))$ =true, and

(2) if $\text{LpQuery}(\Pi, \varepsilon_-(\boldsymbol{c}))$ =true or $\text{LpQuery}(M, \varepsilon_-(\boldsymbol{c}))$ =true, then $\text{LpQuery}(\Pi_M, \varepsilon_-(\boldsymbol{c}))$ =true. Then $\text{PolicyRewrite}(\Pi, M, \Pi_M)$ is complete.

Definition 10 and 11 depict that $\mathbb{S}_M$ absolutely dominates the valid part of the semantic of $\mathbb{S}_\Pi$. Next, we prove that PolicyRewrite is sound and complete under PolicyQuery.

Theorem 7. PolicyRewrite is sound and complete under the algorithm PolicyQuery.

Proof. Soundness is straight forward from Theorem 6.

(1) Suppose $\text{LpQuery}(\Pi, \varepsilon_+(\boldsymbol{c}))$ =true and $\text{LpQuery}(M, \varepsilon_+(\boldsymbol{c}))$ =true. Since $\text{Def}_\Pi(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_\Pi$, and the semantic $\mathbb{S}_\Pi$ of Π decided by LpQuery is supported, $F^+_\Pi|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under $\mathbb{S}_\Pi$. With Π_M as noninterference, we can know that $F^+_\Pi|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under $\mathbb{S}_{\Pi_M}$. Similarly, $F^+_M|_{\boldsymbol{X}=\boldsymbol{c}}$ is also true under $\mathbb{S}_{\Pi_M}$. Hence, $\text{LpQuery}(\mathbb{S}_{\Pi_M}, \varepsilon_+(\boldsymbol{c}))$ =true since $\text{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_+) = \varepsilon_+(\boldsymbol{X}) \leftarrow F^+_M \wedge F^+_\Pi$.

(2) Suppose $\text{LpQuery}(\Pi, \varepsilon_-(\boldsymbol{c}))$ =true. Since $\text{Def}_\Pi(\varepsilon_-) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_\Pi$, and the semantic $\mathbb{S}_\Pi$ of Π decided by LpQuery is supported, then $F^-_\Pi|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under $\mathbb{S}_\Pi$. With Π_M is noninterference, we can know that $F^-_\Pi|_{\boldsymbol{X}=\boldsymbol{c}}$ is true under $\mathbb{S}_{\Pi_M}$. Hence, $\text{LpQuery}(\mathbb{S}_{\Pi_M}, \varepsilon_-(\boldsymbol{c}))$ =true since $\text{Def}^{\mathcal{F}}_{\Pi_M}(\varepsilon_+) = \varepsilon_-(\boldsymbol{X}) \leftarrow F^-_M \vee F^-_\Pi$. Similarly, if $\text{LpQuery}(M, \varepsilon_-(\boldsymbol{c}))$ =true, we also can get $\text{LpQuery}(\mathbb{S}_{\Pi_M}, \varepsilon_-(\boldsymbol{c}))$ =true.

With (1), (2) and Definition 11, we can say that $\text{PolicyRewrite}(\Pi, M, \Pi_M)$ is complete.

Note that PolicyRewrite is not only sound and complete, but also conflict and unsettlement-resolved under $\text{LpQuery}(\Pi_M, \varepsilon_{++}(\boldsymbol{c}))$, which is depicted by Theorem 8.

Theorem 8. (1) $\text{LpQuery}(\Pi, \varepsilon_{++}(\boldsymbol{c}))$ =true and $\text{LpQuery}(M, \varepsilon_{++}(\boldsymbol{c}))$ =true iff $\text{LpQuery}(\Pi_M, \varepsilon_{++}(\boldsymbol{c}))$ =true; (2) $\text{LpQuery}(\Pi_M, \varepsilon_{--}(\boldsymbol{c}))$ =true iff $\text{LpQuery}(\Pi, \varepsilon_{--}(\boldsymbol{c}))$ =true or $\text{LpQuery}(M, \varepsilon_{--}(\boldsymbol{c}))$ =true.

CRB	cr_0 : $\text{label}_{--}(X_e, X_l, X_m) \leftarrow \text{label}_{+}(X_e, X_l, X_m), \text{label}_{-}(X_e, X_l, X_m)$.
SCB	dr_1 : $\text{label}_{+}(X_e, X_l, \text{A}) \leftarrow X_l = 0$. dr_2 : $\text{label}_{+}(X_e, X_l, \text{A}) \leftarrow X_l = 1$. dr_3 : $\text{label}_{+}(X_e, X_l, \text{B}) \leftarrow X_l = 2$. dr_4 : $\text{label}_{+}(X_e, X_l, \text{B}) \leftarrow X_l = 3$.
URB	ur_0 : $\text{label}_{--}(X_e, X_l, X_m) \leftarrow \text{not label}_{+}(X_e, X_l, X_m)$.

Figure 7 BLP-DP (a BLP delegation policy).

CRB	cr_0 : $\text{label}_{--}(X_e, X_l, X_m) \leftarrow \text{label}_{+}(X_e, X_l, X_m), \text{label}_{-}(X_e, X_l, X_m)$.
SCB	pr_1 : $\text{label}_{+}(u_1, 1, \text{A})$. pr_3 : $\text{label}_{+}(X, Y, \text{A}) \leftarrow \text{Trusted}(X, Y, \text{A})$. pr_2 : $\text{label}_{+}(o_1, 2, \text{A})$.
SIB	pr_4 : $\text{Trusted}(u_2, 0, \text{A})$. pr_5 : $\text{Trusted}(u_3, 1, \text{A})$. pr_6 : $\text{Trusted}(u_4, 3, \text{A})$. pr_7 : $\text{Trusted}(X, 0, \text{A}) \leftarrow \text{not Trusted}(X, Y, Z)$. pr_8 : $\text{User}(u_1)$. ⋮ ⋮ pr_{12} : $\text{User}(u_5)$.
URB	ur_0 : $\text{label}_{--}(X_e, X_l, X_m) \leftarrow \text{not label}_{+}(X_e, X_l, X_m)$.

Figure 8 BLP-SP (a BLP security policy).

Proof. Since Π, M and Π_M have the same CRB and URB, the definitions of ε_{++} are the same in them. Suppose $\text{Def}_{\Pi}(\varepsilon_{++}) = \varepsilon_{++}(\boldsymbol{X}) \leftarrow F^{++}$; $\text{Def}_{\Pi}(\varepsilon_{--}) = \varepsilon_{--}(\boldsymbol{X}) \leftarrow F^{--}$; $\text{Def}_{M}(\varepsilon_{++}) = \varepsilon_{++}(\boldsymbol{X}) \leftarrow F^{++}$; $\text{Def}_{M}(\varepsilon_{--}) = \varepsilon_{--}(\boldsymbol{X}) \leftarrow F^{--}$; $\text{Def}_{\Pi_M}(\varepsilon_{++}) = \varepsilon_{++}(\boldsymbol{X}) \leftarrow F^{++} \wedge F^{++}$, $\text{Def}_{\Pi_M}(\varepsilon_{--}) = \varepsilon_{--}(\boldsymbol{X}) \leftarrow F^{--} \vee F^{--}$. Since Π, M and Π_M are noninterference, the truths of F^{++} and F^{--} are decided by SIB. Hence, for any $\boldsymbol{c}$, $F^{++}|_{\boldsymbol{X}=\boldsymbol{c}}$ is true in $\mathbb{S}_{\Pi_M}$ if and only if $F^{++}|_{\boldsymbol{X}=\boldsymbol{c}}$ is true in $\mathbb{S}_{\Pi}$ and $\mathbb{S}_{M}$. With $\mathbb{S}_{\Pi}$, $\mathbb{S}_{M}$ and $\mathbb{S}_{\Pi_M}$ are supported, $\varepsilon_{++}(\boldsymbol{c}) \in \mathbb{S}_{\Pi_M}$ if and only if $\varepsilon_{++}(\boldsymbol{c}) \in \mathbb{S}_{M}$ and $\varepsilon_{++}(\boldsymbol{c}) \in \mathbb{S}_{\Pi}$. We get the proof of (1). Similarly, we can get the proof of (2).

5 A complete example

Formally, delegation policy 2 shown in Figure 6 can be declared as in Figure 7 in our framework, where $\text{label}_{+}(X_e, X_l, X_m)$ denotes that administrator X_m labels subject/object X_e with security level X_l. We can see that delegation policy 2 defines the maximal set of privileges (i.e., labels) that can be authorized to other subject by the intended delegatees (Essentially, labeling a subject or object is an authorization).

Respectively, security policy 2 is formally declared as in Figure 8, where $\text{Trusted}(X, Y, Z)$ denotes that administrator Z scores the reputation of user X with number Y. Obviously, in security policy BLP-SP, A labels u_4 with 3 and labels o_1 with 2. This violates BLP-SP's intended delegation policy BLP-DP. That's to say, there is incompliance in BLP-SP.

In the follow, we illustrate how to resolve the incompliance between BLP-SP and BLP-DP by our approach in detail. With BLP-SP, we can obtain $\text{Def}_{\text{BLP-SP}}(\text{label}_{+})$ of the form

$$\begin{aligned}\text{label}_{+}(X_e, X_l, X_m) \leftarrow {} & (X_e = u_1 \wedge X_l = 1 \wedge X_m = \text{A}) \vee (X_e = o_1 \wedge X_l = 2 \wedge X_m = \text{A}) \vee \\ & \exists_{X,Y}(X_e = X \wedge X_l = Y \wedge X_m = \text{A} \wedge \text{Trusted}(X, Y, A)).\end{aligned}$$

With BLP-DP, we can obtain $\text{Def}_{\text{BLP-DP}}(\text{label}_{+})$ of the form

$$\text{label}_{+}(X_e, X_l, X_m) \leftarrow (X_l = 0 \wedge X_m = \text{A}) \vee (X_l = 1 \wedge X_m = \text{A}) \vee (X_l = 2 \wedge X_m = \text{B}) \vee (X_l = 3 \wedge X_m = \text{B}).$$

With $\text{Def}_{\text{BLP-SP}}(\text{label}_{+})$ and $\text{Def}_{\text{BLP-DP}}(\text{label}_{+})$, we can obtain $\text{Def}_{\text{BLP-SP}_{\text{BLP-DP}}}(\text{label}_{+})$ of the form

$$\begin{aligned}\text{label}_{+}(X_e, X_l, X_m) \leftarrow {} & (X_l = 0 \wedge X_m = \text{A} \wedge \text{Trusted}(X_e, X_l, \text{A})) \vee (X_l = 1 \wedge X_m = \text{A} \wedge X_e = u_1) \\ & \vee (X_l = 1 \wedge X_m = \text{A} \wedge \text{Trusted}(X_e, X_l, \text{A})).\end{aligned}$$

Finally, we simplify the last formula and transform it into a set of logic program rules of the form

$$\{\text{label}_+(X_e,0,\text{A}) \leftarrow \text{Trusted}(X_e,0,\text{A}). \quad \text{label}_+(u_1,1,\text{A}). \quad \text{label}_+(X_e,1,\text{A}) \leftarrow \text{Trusted}(X_e,1,\text{A}).\},$$

Hence, we can get

$$\text{BLP-SP}_{\text{BLP-DP}} = \left\{ \begin{array}{ll} \text{label}_+(X_e,0,\text{A}) \leftarrow \text{Trusted}(X_e,0,\text{A}), & \text{Trusted}(u_2,\text{A}). \\ \text{label}_+(X_e,1,\text{A}) \leftarrow \text{Trusted}(X_e,1,\text{A}), & \text{Trusted}(u_3,\text{A}). \\ \text{label}_+(u_1,1,\text{A}), & \text{Trusted}(u_4,\text{A}). \\ \text{Trusted}(X,0,\text{A}) \leftarrow \text{not Trusted}(X,Y,Z), & \text{User}(u_1),\ldots,\text{User}(u_5). \end{array} \right\},$$

We can see that BLP-SP$_{\text{BLP-DP}}$ resolves all the invalid part but reserves all the valid part of the semantic of BLP-SP with its intended delegation policy BLP-DP. BLP-SP$_{\text{BLP-DP}}$ is readable because it expressed by the same security policy language as BLP-SP and BLP-DP have done.

6 Conclusion

As we know, the core of a delegation policy is to constraint its intended security policies. In order to do so, we propose a new approach which automatically rewrites a given security policy into a new one which is compliant with its intended delegation policy. To our best acknowledge, this is a novel idea which is never proposed in any other works in security policy research fields.

Firstly, we formally define a logic program based security policy language framework, which can resolve all the conflicts and the unsettlements in a security policy (Algorithm 1, Theorem 1 and Theorem 2), and which is sound and complete if the underling logic program query evaluation algorithm is also (Theorem 3).

Then, based on the framework, we propose the rewriting algorithm PolicyRewrite shown in Algorithm 2. Given security policy Π and delegation policy M, PolicyRewrite can transform Π into a new one Π_M which is compliant with M (Theorem 6). Π, M and Π_M are declared in the same style. PolicyRewrite is complete and sound if Π and M are noninterference, and have the same SIB, CRB and URB (Theorem 7). That's to say, Π_M not only resolves all compliance in Π with M but also reserves all the valid part of the semantic of Π with M. Furthermore, Algorithm 1 still works well on Π_M (Theorem 8).

As the first step in compliance maintenance through rewriting approach, in this paper, we require Π and M should be noninterference. This is a strong condition. In future, it is worthwhile to find whether there exist other weak conditions under which we could propose more versatile rewriting algorithms.

Acknowledgements

This work was supported by National Basic Research Program of China (Grant No. 2007CB311100), National High Technology Research and Development Program of China (Grant Nos. 2009AA01Z438, 2009AA01Z431), and National Natural Science Foundation of China (Grant No. 61070186).

References

1 Sandhu R, Coyne E J, Feinstein H L, et al. Role-based access control models. IEEE Comput, 1996, 29: 38–47

2 Schneider F B. Enforceable security policies. ACM Trans Inform Syst Secur, 2000, 3: 30–50

3 Sarna-Starosta B, Stoller S D. Policy analysis for security-enhanced Linux. In: Proceedings of the 2004 Workshop on Issues in the Theory of Security. New York: ACM Press, 2004. 1–12

4 Kikuchi S, Tsuchiya S, Adachi M, et al. Policy verification and validation framework based on model checking approach. In: 4th International Conference on Autonomic Computing (ICAC'07). Washington DC: IEEE Computer Society, 2007. 1–9

5 Anderson J P. Computer Security Technology Planning Study. Technical Report ESD-TR-73-51. ESD/AFSC, Hanscom AFB, Bedford, MA [NTIS AD-758 206]. Volumes I and II, 1972

6 King G, Smith W. An alternative implementation of the reference monitor concept. In: 4th Aerospace Computer Security Applications Conference. Washington DC: IEEE Computer Society, 1988. 159–166

7 Dougherty D J, Fisler K, Krishnamurthi S. Specifying and reasoning about dynamic access-control policies. In: IJCAR'06: International Joint Conference on Automated Reasoning. New York: Springer, 2006. 632–646

8 Haudhuri A, Naldurg P, Rajamani S K, et al. EON: modeling and analyzing dynamic access control systems with logic programs. In: Proceedings of the 15th ACM Conference on Computer and Communications Security. New York: ACM, 2008. 27-31, 381–390

9 Harrison M A, Ruzzo W L, Ullman J D. Protection in operating systems. Commun ACM, 1976, 19: 461–471

10 Wallace D R, Fujii R U. Software verification and validation: an overview. IEEE Software, 1989, 6: 10–17

11 Clarke E M, Emerson E A, Sistla A P. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans Progr Lang Sys, 1986, 8: 244

12 Holzmann G J. The model checker SPIN. IEEE Trans Software Eng, 1997, 23: 279–295

13 LaPadula L J, Bell D E. Secure Computer Systems: a Mathematical Model. Technical Report ESD-TR-278, Vol 2. MA: The Mitre Corp., 1973

14 Woo T Y C, Lam S S. Authorizations in distributed systems: a new approach. J Comput Secur, 1993, 2: 107–136

15 Jajodia S, Samarati P, Sapino M L, et al. Flexible support for multiple access control policies. ACM Trans Database Syst, 2001, 26: 214–260

16 Gurevich Y, Neeman I. DKAL: distributed-knowledge authorization language. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium. Washington DC: IEEE Computer Society, 2008. 149–162

17 Becker M Y, Sewell P. Cassandra: distributed access control policies with tunable expressiveness. In: 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004). Washington DC: IEEE Computer Society, 2004. 159–168

18 DeTreville J. Binder, a logic-based security language. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. Washington DC: IEEE Computer Society, 2002. 105–113

19 Trevor J. Sd3: a trust management system with certified evaluation. In: IEEE Symposium on Security and Privacy. Washington DC: IEEE Computer Society, 2001. 106–115

20 Lloyd J W. Foundations of Logic Program. 2nd ed. New York: Springer-Verlag, 1987. 17, 78–79

21 Damianou N, Dulay N, Lupu E, et al. The ponder policy specification language. In: Proceedings of the International Workshop on Policies For Distributed Systems and Networks (Policy'01). LNCS. London: Springer-Verlag, 2001. 18–38

22 Apt K R, Blair H A, Walker A. Towards a theory of declarative knowledge. In: Foundations of Deductive Databases and Logic Programming. San Francisco: Kaufmann Publishers Inc., 1988. 89–148

23 Gelfond M, Lifschitz V. The stable model memantics for logic programming. In: Proceedings of ICLP'88. Seattle: MIT Press, 1988. 1070–1080

24 Gelder A V, Ross K A, Schlipf J S. The well-founded semantics for general logic programs. J ACM, 1991, 38: 619–649

25 Niemela I, Simons P. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In: Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning. London: Springer-Verlag, 1997. 421–430