Trust Negotiation with Nonmonotonic Access Policies

Phan Minh Dung and Phan Minh Thang

Department of Computer Science, Asian Institute of Technology,
GPO Box 4, Klong Luang, Pathumthani 12120, Thailand
{dung, thangphm}@cs.ait.ac.th

Abstract. We study the structure of nonmonotonic access policies for internet-based resources. We argue that such policies could be divided into two parts: the locally designed policies and imported policies. Imported policies should always be monotonic while the local policies could be nonmonotonic. We develop a safe proof procedure for nonmonotonic trust negotiation where safety means that access to a resource is granted only if its access policy is satisfied.

1 Introduction

Blaze, Feigenbaum and Lacy [1] introduced trust management (TM) as a new approach to decentralized authorization. An access decision in TM is based on two sources of information obtained from the credentials submitted by the clients and from local databases of collected credentials and observations. An example is an access policy of an auction site stating that a client with a valid digital credit card and no record of cheating is allowed to participate in its auction service. Such rule could be represented using Horn clauses as follows:

$$\text{Believe(S,TrustWorthy(Auction),C)} \leftarrow \text{Believe(S,HaveFund,C)}, \text{ not Believe(S,Fraudster,C)}$$

stating that server S believes that client C is trustworthy for access to the auction if S believes that C has sufficient fund and S has no evidence to believe that C is a fraudster where a valid credit card is a convincing proof for S that the client has sufficient fund.

It has been recognized in the literature that one of the key requirements for TM access policies is that it should be monotonic with respect to the client's submitted credentials but could be nonmonotonic with respect to the site's local information about the client [12]. This requirement is designed to avoid situations in which the client has been given access to some services, but later when he submits new credentials for other services, and the disclosure of the new credentials may terminate the access to those services granted to him before. The question of what kind of structure access policies should have to satisfy this requirement is still open.

A. Aagesen et al. (Eds.): INTELLCOMM 2004, LNCS 3283, pp. 70–84, 2004.

A key aspect in TM is delegation. Delegation allows a principal to transfer authority over some resources to other principals. Delegation hence divides a principal's access policies into two parts: The principal's own policies and other components that are imported. Consider for example the policies of a book store that offered discount to its preferred customer [10]. Students from a nearby university U are its preferred customers. The book store policy also states that any preferred customer of an E-organization is also its preferred customer. The access policy hence consists of two parts: the book store local regulation that directly identifies who gets discount, and the imported regulation of the E-organization about its preferred customers.

Imported policies are rules to determine the beliefs of those who issued them. Therefore imported policies should be monotonic as otherwise, to evaluate them, an agent would need to have access to the entire information base (often including sensitive information) of the issuers of such policies. However in practice, agents are unlikely to let other agents having access to their sensitive local information. Hence, it is natural to expect imported policies to be monotonic.

Herzberg et all [6] has discussed nomonotonicity for access policies without imported rules. The monotonicity with respect to the client submitted credentials was not discussed in [6]. Though a proof procedure for nonmonotonic access policies has been given in [6], it is not clear what kind of declarative semantics this procedure has and especially how it is related to the semantics of nonmonotonic reasoning.

Trust negotiation is a process of exchanging certificates and policy statements that allows one party to establish sufficient trust on the other party to allow it access to some resource.

Logic programming has been shown to be an appropriate framework for studying trust management [10]. It is also well-known that the mechanism of negation as failure in logic programming provides a powerful tool for nonmonotonic reasoning [4, 5].

In this paper we study the structure of nonmonotonic access policies and develop a procedure for trust negotiation with nonmonotonic access policies. Our procedure is based on the sldnf procedure in logic programming. We then show that the proposed procedure produces safe negotiation in the sense that access to a resource is granted only if its access policy is satisfied.

2 Preliminaries: Logic Programming and Stable Model Semantics

A *program clause* is of the form $a \leftarrow a_1, \ldots, a_n, not\, b_1, \ldots, not\, b_m$ where a, $a_1, \ldots, a_n, b_1, \ldots, b_m$ are atoms. The clause is called *definite if* $m = 0$. A *logic program* is a set of program clauses.

Let P be a logic program and G be the set of all the ground instances of clauses in P. A *stable model* of P is defined as a set of ground atoms M such that M is the least Herbrand model of P_M where P_M is obtained from G as follows:

Delete every clause C from G whose body contains a negative literal $notA$ such that $A \in M$.

Delete all negative literals from the remaining clauses.

We write $P \models A$ for a ground atom A if A belongs to all stable models of P. More about semantics of logic programs could be found in [4, 5].

3 Structure of Nonmonotonic Access Policies

We assume an alphabet consisting of the following components:

- A set $\mathcal{R}$ of role (also called attribute) names
- A set of principal identifiers PI
- A set RE of resource identifiers.
- A distinct unary attribute symbol *Trustworthy* (often abbreviated as TW)
- A ternary predicate *Bel(x,R,y)* stating that x believes that y has attribute R.
- A binary predicate symbol *Hold(R,x)* stating that x has attribute R.

A *principal term* is either a principal identifier from PI or a PI-variable where a PI-variable is a variable that could be instantiated with values from PI only.

A *certificate* is of the form Cert(A,R,B) where A,B are principal identifiers from PI and R is an attribute term. The purpose of a certificate is to certify that A believes that B has the attribute R. In practice, certificates have more complex structures. We restrict ourself on a simple form of certificates as we are focused on the study of nonmonotonic access policies. Certificates represent an important kind of resources that are different from those resources represented by resource identifiers from RE. We define a *resource term* either as a resource identifier, or a certificate.

A *attribute term* has the form $R(t_1, \ldots, t_n)$ where R is an n-ary attribute symbol from $\mathcal{R}$ and $t_1, \ldots, t_n$ are resource terms.

An *atom* is either of the form Bel(p,T,q) or Hold(T,p) where p,q are principal terms and T is an attribute term. q is called the *subject* of the atom while p is its *issuer*. A literal is an atom or the negation of an atom. The subject or issuer of a literal is the subject or issuer of its atom respectively.

Let $\mathcal{S}$ be a set of belief atoms and x,y be two principal terms appearing in some atoms in S. We say that there is a **flow of trust** from x to y in $\mathcal{S}$ if there are principal terms $p_1, \ldots, p_m$ and attribute terms $T_1, \ldots, T_{m-1}$ such that $Bel(p_i, T_i, p_{i+1}) \in \mathcal{S}$ and x $= p_1$ and $y = p_m$.

A *policy clause* of a principal A is of the form:

$$Bel(A, T, p) \leftarrow \alpha_1, \ldots, \alpha_n, not\, \alpha_{n+1}, \ldots, not\, \alpha_{n+k}$$

where A is a principal identifier, p is a principal term, T is a attribute term and $\alpha_1, \ldots, \alpha_{n+k}$ are atoms such that every variable except p appears as the subject of some positive literal in the body of the clause. The intuition behind this condition is that there is a flow of trust from some well-known principals, represented as principal identifiers in the clause, to any principal that could possibly appear in A's policy. A is called the *issuer* of the clause.

A principal term p is said to be *redundant* in a policy clause if there exists no flow of trust from p to the subject of the head of the clause in the set of positive literals of the clause body. A policy clause is said to be *nonredundant* if there is no redundant principal terms in its body.

It is not difficult to see that credentials defined in the languages RT_0, RT_1, RT_2 in the RT family [10] could be represented either as a certificate or as a policy clause in our framework.

An *access policy* of an principal A is defined as a a pair $APL = (LPL, IPL)$ where

- LPL is a finite set of local nonredundant policy clauses of A.
- IPL is a finite set of imported nonredundant policy clauses whose issuers are not A.

Consider the access policies of the book store (BS) example in the introduction. The policy clauses of BS are the following:

$$Bel(BS, TW(Discount), x) \leftarrow Bel(BS, PreferredCustomer, x)$$
$$Bel(BS, PreferredCustomer, x) \leftarrow Bel(U, Student, x)$$
$$Bel(BS, PreferredCustomer, x) \leftarrow Bel(EOrg, PreferredCustomer, x)$$

while the imported clauses are those determining who are the preferred customers of EOrg.

Imported policies are rules to determine the beliefs of those who issued them. Therefore imported policies should be monotonic as otherwise, to evaluate them, an agent would need to have access to the entire information base (often including sensitive information) of the issuers of such policies. However in practice, agents are unlikely to let other agents having access to their sensitive local information. Hence, it is natural to expect imported policies to be monotonic.

The *attribute dependency graph* of a access policy P is a directed graph whose nodes are the attributes appearing in P, and there is a positive (resp. negative) edge from α to β if α appears in the head of a clause in P and β appears in positive (resp. negative) literal in its body.

A path in the attribute dependency graph of P is said to be *positive* (resp *negative*) if all (resp. some) edges on this path are positive (resp. negative).

Now we can define formally the notion of a trust management system.

Definition 1. *Let A be a principal identifier. A Trust Management System (TMS) for A is represented as a quadruple* $\langle APL, DBO, DBC, CA\rangle$ *consisting of*

1. *An access policy* $APL = (LPL, IPL)$ *of A such that all imported clauses in it are definite.*
2. *a set DBO of ground atoms of the form Hold(R,B) where R is a ground attribute term and B is a principal identifier. Atoms in DBO represent information A has collected locally about other principals.*
3. *a set of certificates DBC that are in A's possession.*

4. *a set of client attributes $CA \subseteq \mathcal{R}$ that the A expects the client to satisfy. CA is hence required to satisfy the following conditions:*
   (a) *For each $T \in CA$, T does not appear in the head of each of the clauses of APL.*
   (b) *All paths leading to attributes in CA in the attribute dependency graph of P are positive.*
   As we will see shortly this condition ensures that the access policy is monotonic with respect to the client's submitted credentials.

From definition 1, it follows immediately that there is no path linking an attribute that appears in a negative literal in the body of some clause of APL to an attribute in CA in the attribute dependency graph of APL. This condition guarantees that when a server checks a negative condition, it does not require the client to send extra information.

Example 1. Consider the trust management system $\langle APL, DBO, DBC, CA\rangle$ of an agent S who oversees the access to sensitive documents in a hospital. The policy states that only doctors who could present a credential from a recognized hospital and are not known to have a careless conviction from recognized hospitals, have access to the documents. A recognized hospitals is either known locally or certified by other recognized hospitals [6]. The hospital access policies could be expressed as follows:

$$Bel(S, TrustWorthy(R), x) \leftarrow not\ Bel(S, Convicted, x),\ Bel(y, Doctor, x), \\ Bel(S, RecognizedHospital, y)$$
$$Bel(S, RecognizedHospital, x) \leftarrow Hold(RecognizedHospital, x)$$
$$Bel(S, RecognizedHospital, x) \leftarrow Bel(S, RecognizedHospital, y), \\ Bel(y, RecognizedHospital, x)$$
$$Bel(S, Convicted, x) \leftarrow Bel(S, RecognizedHospital, y), Bel(y, Convicted, x),$$

where R denotes the sensitive documents.

The local certificate database DBC consists of certificates Cert(S,Recognized Hospital,H), Cert(H,RecognizedHospital,K) and Cert(H,Convicted,P). The local database DBO contains the fact Hold(RecognizedHospital,H). The set of client attributes CA is defined by $CA = \{Doctor\}$.

Definition 2. *Let C be a principal identifier and $\mathcal{A} = \langle APL, DBO, DBC, CA\rangle$ be a TMS. A set SC of basic credentials of the form Cert(B,T,C) with $T \in CA$ is said to be a guarantee for C to get access to a resource R wrt $\mathcal{A}$ if*

$$APL \cup DBO \cup Th \models Bel(A, TrustWorthy(R), C)$$

where $Th = \{Bel(B, S, D) \mid Cert(B, S, D) \in DBC \cup SC\}$

The monotonicity with respect to the client submitted credentials is stated in the theorem below.

Theorem 1. *Let $\mathcal{A} = \langle APL, DBO, DBC, CA\rangle$ be a TMS of A, C be principal identifiers, SC be a guarantee for C to get access to R wrt $\mathcal{A}$ and SC' be a set of credentials of the form Cert(B,T,C) with $T \in CA$ such that $SC \subseteq SC'$. Then SC' is also a guarantee for C to get access to R wrt $\mathcal{A}$.*

Proof. Let $P = APL \cup DBO \cup \{Bel(B,S,D) \,|\, Cert(B,S,D) \in DBC \cup SC\}$ and $P' = APL \cup DBO \cup \{Bel(B,S,D) \,|\, Cert(B,S,D) \in DBC \cup SC'\}$. Further let $SC_0 = SC' \setminus SC$. Further let M' be stable models of P'. It is not difficult to see that $P'_{M'} = P_{M'} \cup \{Bel(B,S,D) \,|\, Cert(B,S,D) \in SC_0\}$. Let M be the least Herbrand model of $P_{M'}$. Hence $M \subseteq M'$. It is not difficult to see that for each atom $\alpha \in M' \setminus M$, there is a positive path from the attribute of α to an attribute of a certificate in SC_0 in the attribute dependency graph of APL. From the structure of trust management system (definition 1), it follows that α does not appear as a ground instance of a negative literals in any of the policy clauses. Hence $P_M = P_{M'}$. Hence M is a stable model of P. From the assumption that SC be a guarantee for C to get access to R wrt $\mathcal{A}$, it follows immmediately $Bel(A, TW(R), C) \in M'$. The theorem is proved.

4 Trust Negotiation with Nonmonotonic Access Policies

When a principal A wants to access a resource R controlled by B, A sends a request to B. B will consult its local policy to check whether A is trustworthy enough to be given access to R. During this process, B may ask A to send over some certificates to certify certain attributes of A. If the checking process is successful, B will send A a message informing it that its request for access to R has been granted. On the other hand, when A gets requests from B for A's certificates, A consults its own local policy to check whether B should be given access to the requested certificates. A may ask B to send over some certificates before sending B the requested certificates. An example is a scenario in which a client of a E-business orders some good. The business may ask the client for a credit card. Before sending the credit card to the business, the client may ask for a Better Business Bureau certificate from the business. In the following, we will model these processes.

There are many possible strategies on how trust negotiation could be conducted. Consider an example of a policy governing access to sensitive documents of a top secret project where only members of partner projects are allowed to access the documents.

$$Bel(S, TrustWorthy, x) \leftarrow Hold(Partner, y), Bel(y, Member, x),$$

An agent could work on many projects and is reluctant on its part to disclose its associations to these projects.

When getting a access request, S could reveal the partner projects and asks the client to prove its association to one of them. This would reveal sensitive information about identity of the partner projects and hence unacceptable to S. S could on the other hand ask the client to identify the projects he works in.

If one of them is a partner project of S, access is granted for the client. This would force the client to reveal its association to projects that it may consider to be sensitive. Which one is preferred could hardly be determined without considering the real context of such applications. The example indicates that there may be no conceptually best access policies evaluation strategy for all participants involved. The evaluation proof procedure we are going to present shortly may be an appropriate one in one context and less so in others. But anyway it represents an option that needs to be taken into consideration when a method is designed for access policy evaluation in an application.

The negotiation strategy developed in this paper is biased toward the manager of a resource. In the above example, when getting a access request, the server asks the client for credentials certifying its association to projects he works in. In this way, the server could protect its data but the client may have to expose more sensitive information than it loves to.

There are two kinds of requests that principals may send to each other:

- Original requests that start a negotiation process:

$$A \textbf{ to } B : \ Bel(B, TW(R), A)$$

  stating intuitively that A (the sender) asks B (the receiver) to check whether A is trustworthy for access to R.
- Requests that are sent in response to an earlier request:

$$A \textbf{ to } B : \ Bel(x, T, B)$$

  stating intuitively that A asks B for certificates certifying that B has attribute T.

Negotiation results are sent in messages of the following form:

$$A \textbf{ to } B : \ \textbf{success(R)}$$
$$A \textbf{ to } B : \ fail$$

in which A informs B that the negotiation for access to R has succeeded or failed respectively.

During a trust negotiation, the sets of certificates collected by participants change as the principals involved may have to send to the other side a number of certificates. We define a *state* of a principal B during a negotiation as a pair (sc,ss) where sc represents the set of certificates it has collected so far in his database of certificates and ss represents the set of certificates it has sent to the other side from the start of the current negotiation until now.

A negotiation is characterized by state change caused by sending and receiving requests. We use the notation $(sc, ss) \xrightarrow{M?;N!}_B (sc', ss')$ (resp. $(sc, ss) \xrightarrow{M!;N?}_B (sc', ss')$) to denote that when B receives (resp. sends) a request M, B will start its part in a negotiation process to satisfy M and B ends the negotiation when B sends out (resp. receives) message N containing the result of the negotiation. At the end of the negotiation, sc" is the set of credentials B has collected so far and ss" is the set of credentials B has sent over to A.

Definition 3. *Suppose principals A,B are in a state* $st = (sc, ss)$, $st' = (sc', ss')$. *A state transition is triggered when a request M is sent or received.*

1. *Let M be of the form*

$$A \textbf{ to } B : \ Bel(B, TW(R), A)$$

where R is a resource but not a certificate. A negotiation is initiated when M is sent from A to B. It follows that $ss = ss' = \emptyset$. *When B receives M, B checks its access policy to see whether A is trustworthy for access to R. Formally B constructs a local derivation (to be defined shortly) of the form*

$$Ld = (G_0, sc, \emptyset), \ldots, (G, sc", ss")$$

and $G_0 = Bel(B, TW(R), A)$.

(a) If Ld is a successful local derivation wrt B (to be defined shortly) then following transition happens

$$(sc, \emptyset) \xrightarrow{M?;N!}_B (sc", ss")$$

$$(sc', \emptyset) \xrightarrow{M!;N?}_A (sc' \cup ss", sc" \setminus sc)$$

where N has the form

$$B \textbf{ to } A : \ \ success(R)$$

(b) If Ld is a failed local derivation wrt B (to be defined shortly) then following transition happens

$$(sc, \emptyset) \xrightarrow{M?;N!}_B (sc", ss")$$

$$(sc', \emptyset) \xrightarrow{M!;N?}_A (sc' \cup ss", sc" \setminus sc)$$

where N has the form

$$B \textbf{ to } A : \ fail$$

2. *Let M be of the form*

$$A \textbf{ to } B : \ Bel(p, T, B)$$

stating that A needs access to some certificate certifying that B has property T. Note that p is a principle term. Upon receiving M, B will check for those certificates of the form Cert(C,T,B) in its pool of certificate DBC_B. *B selects one of them and consults its local policy to check whether A could be given access to it. If the check is successful, the certificate will be sent to A If the check fails another certificate of the form Cert(C,T,B) is selected and check whether it could be sent to A. The process continues until either B finds a certificate to send to A or B breaks the negotiation by sending a fail message to A. This process is formalized as follows:*

Let $SC = \{C_1, \ldots, C_m\}$, $m \geq 0$ *be the set of certificates in SC of the form* $Cert(C_i, T, B)$ *such that p,* C_i *are unifiable.*

(a) If $SC = \emptyset$ then following transition happens:

$$(sc, ss) \xrightarrow{M?;N!}_B (sc, ss)$$

$$(sc', ss') \xrightarrow{M!;N?}_A (sc', ss')$$

where N has the form

$$B \textbf{ to } A: \ fail$$

(b) Let $SC \neq \emptyset$. Let $G_0 = K_1 \vee \ldots \vee K_m$ where $K_i = Bel(B, TW(C_i), A)$. There are two cases:

i. There is a successful local derivation wrt B of the form

$$(G_0, sc, ss), \ldots, (H, sc", ss")$$

with $H = nil \vee K_{i+1} \vee \ldots \vee K_m$. Then following transition happens

$$(sc, ss) \xrightarrow{M?;N!}_B (sc", ss" \cup \{C_i\})$$

$$(sc', ss') \xrightarrow{M!;N?}_A (sc' \cup (ss" \setminus ss) \cup \{C_i\}, ss' \cup (sc" \setminus sc))$$

where N has the form

$$B \textbf{ to } A: \ \ success(C_i)$$

We will see later, a successful local derivation $(G, sc, ss), \ldots, (H, sc', ss')$ wrt B means that B has successively check that A could be given access to some of the certificate in SC. From $H = nil \vee K_{i+1} \vee \ldots \vee K_m$, this certificate is identified as C_i.

ii. There is a failed local derivation wrt B of the form

$$(G_0, sc, ss), \ldots, (\emptyset, sc", ss")$$

then

$$(sc, ss) \xrightarrow{M?;N!}_B (sc", ss")$$

$$(sc', ss') \xrightarrow{M!;N?}_A (sc' \cup (ss" \setminus ss), ss' \cup (sc" \setminus sc))$$

where N has the form

$$B \textbf{ to } A: \ fail$$

We introduce now the notion of local derivation. First we define a *goal* as a disjunction $K_1 \vee \ldots \vee K_n$ where each K_i is a conjunction of literals.

Intuitively a local derivation from a goal G wrt B is a sequence of goals whose first element is G. Each step in the derivation corresponds to the application of some inference rule which replaces one of the conjunctions by a goal. In this paper, we use a depth-first strategy by always selecting the leftmost conjunction for expansion. A derivation is successful if one of the conjunction is an empty one.A derivation is failed if the last goal is the empty disjunction[1]. In the following, we give a formal definition of the inference steps involved.

[1] Note that empty conjunction denotes true while empty disjunction denotes false.

Let $B = \langle APl_B, DBO_B, DBC_B, CA_B \rangle$. Formally, a local derivation wrt B from a goal G is a sequence of pairs $(G_0, st_0), \ldots, (G_n, st_n)$ where G_i are goals, $G_0 = G$, $st_i = (sc_i, ss_i)$ are states of B. Each G_i in the sequence is obtained from the previous one using an inference rule given below. We employ depth-first search strategy by always selecting the leftmost literal in the leftmost conjunction for expansion.

For the purpose of simple reference, we call an atom of the form $Bel(x, T, A)$ where $T \in CA_B$ an *input atom* of B as A is expected to provide a certificate to certify it.

Definition 4. *Let L be the selected atom in G_i and suppose that G_i has the form $K_1 \vee \ldots \vee K_m$, where each K_i is a conjunction of literals. Let $K_1 = LK_1'$*[2]*. $(G_{i+1}, sc_{i+1}, ss_{i+1})$ is obtained from (G_i, sc_i, ss_i) by applying one of the following steps:*

1. *(Unfolding). L is a positive literal that is not an input atom*[3]*. Let $Cl = \{cl_1, \ldots, cl_k\}$ be the set of clauses in*

$$APl_B \cup DBO_B \cup \{Bel(D, S, E) \leftarrow \mid Cert(D, S, E) \in sc_i\}$$

such that the heads of these clauses are unifiable with L and for each i, θ_i is the most general unifier (mgu) of L and the head of cl_i. There are two cases:

(a) Cl is empty. Then

$$G_{i+1} = K_2 \vee \ldots \vee K_m$$

$$(sc_{i+1}, ss_{i+1}) = (sc_i, ss_i)$$

(b) Cl is not empty. Let bd_i be the body of cl_i

$$G_{i+1} = (bd_1 K_1')\theta_1 \vee \ldots \vee (bd_k K_1')\theta_k \vee K_2 \vee \ldots \vee K_n$$

$$(sc_{i+1}, ss_{i+1}) = (sc_i, ss_i)$$

2. *(Negation As Failure). L is a negative literal. There are two cases:*

(a) L is not ground. Then

$$G_{i+1} = K_2 \vee \ldots \vee K_m$$

$$(sc_{i+1}, ss_{i+1}) = (sc_i, ss_i)$$

(b) L is ground. There are two cases:

[2] For simplicity, a conjunction is written as a sequence of its conjuncts.
[3] i.e. L has the form $Bel(p, T, C)$ such that $T \notin CR_B$.

i. $L = not\, Bel(B, T, D)$.

If there is a failed local derivation wrt B from $(Bel(B, T, D), sc_i, ss_i)$ *then*

$$G_{i+1} = K'_1 \vee K_2 \vee \ldots \vee K_m$$

$$(sc_{i+1}, ss_{i+1}) = (sc_i, ss_i)^4$$

If there is successful local derivation wrt B from $(Bel(B, T, D), sc_i, ss_i)$ *then*

$$G_{i+1} = K_2 \vee \ldots \vee K_m$$

$$(sc_{i+1}, ss_{i+1}) = (sc_i, ss_i)$$

ii. $L = not\, Hold(T, C)$.

If $Hold(T, C) \notin DBO_B$ *then*

$$G_{i+1} = K'_1 \vee K_2 \vee \ldots \vee K_m$$

$$(sc_{i+1}, ss_{i+1}) = (sc_i, ss_i)$$

If $Hold(T, C) \in DBO_B$ *then*

$$G_{i+1} = K_2 \vee \ldots \vee K_m$$

$$(sc_{i+1}, ss_{i+1}) = (sc_i, ss_i)$$

3. (Asking for Credential). L is a positive input literal, i.e L has the form $Bel(p, T, A)$ *with* $T \in CA_B$ *and p a (possibly nonground) principal term.*

Let $SC = \{C_1, \ldots, C_k\}$, $m \geq 0$ *be the set of credentials in* sc_i *of the form* $Cert(C_i, T, A)$ *and* θ_i *be the substitution* $\{p/C_i\}$ *assigning* C_i *to p. There are two cases:*

(a) $SC \neq \emptyset$. *Then*

$$G_{i+1} = K_{1,1} \vee \ldots \vee K_{1,k} \vee K_2 \vee \ldots \vee K_m$$

$$(sc_{i+1}, ss_{i+1}) = (sc_i, ss_i)$$

where $K_{1,j} = K'_1\theta_j$

[4] Note that due to lemma 1, the sets sc_i, ss_i do not change in any local derivation of Bel(B,T,D).

(b) $SC = \emptyset$*, i.e. B can not find any certificate in its pool that certifies the belief L. B then starts a negotiation by sending A a request M of the form*

$$B \textbf{ to } A : \ Bel(p, T, A)$$

If there is a successful negotiation of B with A represented by a transition $(sc_i, ss_i) \stackrel{M!;N?}{\longrightarrow}_B (sc, ss)$ *where N is a success message of the form "A* **to** *B: success(C)", then*

$$G_{i+1} = K_1'\theta \vee K_2 \vee \ldots \vee K_k$$

if p is a variable and θ *is the substitution* $\{p/D\}$ *assigning D to p and* $C = Cert(D, T, A)$*. Otherwise*

$$G_{i+1} = K_1 \vee K_2 \vee \ldots \vee K_k$$

In both cases

$$(sc_{i+1}, ss_{i+1}) = (sc, ss)$$

If there is a failed negotiation of B with A represented by $(sc_i, ss_i) \stackrel{M!;N?}{\longrightarrow}_B (sc, ss)$ *where N is a fail message of the form A* **to** *B: fail, then*

$$G_{i+1} = K_2 \vee \ldots \vee K_k$$

$$(sc_{i+1}, ss_{i+1}) = (sc, ss)$$

A local derivation $(G_0, sc_0, ss_0), \ldots, (G_n, sc_n, ss_n)$ of B is *successful* if G_n is of the form $nil \vee \mathcal{D}$. It *fails* if G_n is an empty disjunction.

Lemma 1. *Let* $B = \langle APl_B, DBO_B, DBC_B, CR_B \rangle$*, and* $sc_0 = DBC_B$*. Let* $(G_0, sc_0, ss_0), \ldots, (G_n, sc_n, ss_n)$ *be a local derivation wrt B with* $G_0 = L$ *such that notL is a negative literal appearing in an ground instance of a policy clause in* APL_B*. Then there are no asking-for-credential-steps in the derivation and* $sc_n = sc_0$ *and* $ss_n = ss_0$*.*

Proof. Obvious from the fact that there is no path from a attribute occuring in a negative literal to an attribute in CA_B in the attribute dependency graph.

Example 2. Consider the hospital example 1. Suppose that P wants to access the sensitive documents. P has a certificate C = Cert(H,Doctor,P) issued by hospital H. P is willing to show every body his certificate, i.e. APL_P consists of the only clause

$$Bel(P, TW(C), x) \leftarrow$$

P starts a negotiation with S by sending S a request M of the form "P **to** S: Bel(S,TW(R),P)". After receiving M, S starts a local derivation as follows

$$Ld = (G_0, sc_0, ss_0),\ (G_1, sc_0, ss_0),\ (G_2, sc_0, ss_0)$$

to check whether P is trustworthy for access to the documents where

$G_0 = Bel(S, TW(R), P)$
$G_1 = not\ Bel(S, Convicted, P), Bel(y, Doctor, P),$
$\quad Bel(S, RecognizedHospital, y)$
$G_2 = \emptyset$ and
$sc_0 = DBC,\ ss_0 = \emptyset.$

Note that the selected subgoal in G_1 is $not\ Bel(S, Convicted, P)$. As there is a successful local derivation from $(Bel(S, Convicted, P), sc_0, \emptyset)$ to $(nil, sc_0, \emptyset)$, we have $G_2 = \emptyset$.

S hence informs P that his request is rejected. We have

$$(sc_0, \emptyset) \xrightarrow{M?;N!}_S (sc_0, \emptyset)$$
$$(\{C\}, \emptyset) \xrightarrow{M!;N?}_P (\{C\}, \emptyset)$$

where N is of the form "S **to** P: fail".

The following theorem shows that the negotiation defined in this chapter is safe in the sense that access to a resource is granted to a client only if it has produces a guarantee to establish its trustworthiness.

Theorem 2. *(Safe Negotiation)*
Let $B = \langle APl_B, DBO_B, DBC_B, CR_B \rangle$, *and* $sc_0 = DBC_B$.

1. *Let* $(G_0, sc_0, ss_0), \ldots, (G_n, sc_n, ss_n)$ *be a local derivation wrt B with* $G_0 = \{Bel(B, TW(R), A)\}$. *Then* $sc_n \backslash sc_0$ *is a guarantee of Bel(B, TrustWorthy(C), A) for each certificate* $C \in ss_n \setminus ss_0$.
   If the derivation is successful then $sc_n \setminus sc_0$ *is a guarantee of Bel(B, TrustWorthy (R),A)*
2. *Suppose that*

$$(sc, ss) \xrightarrow{M?;N!}_B (sc', ss')$$

or

$$(sc, ss) \xrightarrow{M!;N?}_B (sc', ss')$$

where $sc = DBC_B$. *Then for each* $C \in ss' \setminus ss$, $sc' \setminus sc$ *is a guarantee for Bel(B, TW(C),A) wrt B where A is the other party in the negotiation.*

Proof (Sketch). Assertion 2 follows immediately from assertion 1. Assertion 1 is proved by induction on the depth of the nested negotiation invoked in asking-for-credential-steps. The full proof is tedious and long and the readers are referred to the full version of this paper.

5 Conclusion and Related Works

We have studied the structure of nonmonotonic access policies and provided a general sufficient condition that guarantees the monotonicity wrt the client submitted credentials. We also have argued that only locally defined policy clauses should be nonmonotonic. The semantics of our policy language is based on the stable semantics of logic programming. We have also given a procedure for trust negotiation within our framework and showed its safety.

A weakness of our negotiation procedure is that the negotiation parties do not know whether they have submitted enough credentials for access to a resource until access is granted. This problem could be avoided by sending partially evaluated policies instead of requests for certificates like in [3, 13]. We also do not consider the privacy of local data and policies. In the future works, the procedure should be extended to deal with these problems.

Our work is based and inspired by a large body of works on trust management and negotiation [1, 3, 6, 9, 10] though with the exception of Herberg et all [6], no author has studied problems related to nonmonotonic access policies.

Bonatti and Saramanti [3] present a framework for regulating access control and information release. Access policies are monotonic and are represented by condition-action rules. The credentials are complex and represented by terms.

Trust negotiation and strategies have been studied extensively in [9, 13]. Several criteria for trust negotiation have been proposed in [13]. It would be interesting to see how these criteria could be incorporated into our framework.

Our framework is very much inspired by the RT frameworks proposed by Li, Mitchell and Winsborough [10]. Both systems are based on logic programming. While the RT framework is proposed to combine the strengths of role-based access control and trust management, our is focused on the nonmonotonicity of access policies.

References

1. M. Blaze, J. Feigenbaum, J. Lacy Decentralized Trust Management. In *Proc of the 17th IEEE Symposium on Security and Privacy, Oakland, CA, May 1996*
2. M. Blaze, J. Feigenbaum, M. Strauss Compliance Checking in the PolicyMaker Trust management System. In *Proc. of Financial Cryptography '98, LNCS 1465, 1998*
3. P. A. Bonatti, P. Samarati A Uniform Framework for Regulating Service Access and Information Release on the Web. In Conference on Computer and Communication Security, Athens, Greece, 2000
4. P. M. Dung. Negation as hypothesis: an argument-based foundation for logic programming. Journal of Logic Programming, 1994
5. M. Gelfond, V. Lifschitz, The stable model semantics for logic programming. iclp5thWashington, Seattle1988K. Bowen and R. A. Kowalski, eds 1070–1080
6. A. Herzberg, I. Golan, O. Omer, Y. Mass. An efficient algorithm for establishing trust in strangers *http://www.cs.biu.ac.il/ herzbea/Papers/PKI/ec01-paper.pdf*

7. A. Hess, B. Smith, J. Jacobson, K. E. Seamons, M. Winslett, L. Yu, T. Yu. Negotiating Trust on the Web, In *IEEE Internet Computing*, pages 30-37. IEEE Press. November 2002.
8. N. Li, W. H. Winsborough, Towards Practial Automated Trust Negotiation. In *IEEE 3rd Intl. Workshop on Policies for Distributed Systems and Networks (Policy 2002)*. IEEE Press, June 2002.
9. X. Ma, M. Winslett, T. Yu. Prunes: An Efficient and Complete Strategy for Automated Trust Negotiation over the Internet. In *Proceeding of Seventh ACM Conference on Computer and Communications Security*(CCS-7), pages 210-219. ACM Press, November 2000.
10. N. Li, J. C. Mitchell, W. H. Winsborough. Design of a Role-based Trust-management Framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy, May 2002*.
11. J. C. Mitchell, N. Li, W. H. Winsborough, Distributed Credential Chain Discovery in Trust Management. In *Proceeding of Eighth ACM Conference on Computer and Communications Security*(CCS-8), pages 156-165. ACM Press, November 2001.
12. K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, L. Yu. Requirements for Policy Languages for Trust Negotiation. In 3rd International Workshop on Policies for Distributed Systems and Networks, June 2002
13. T. Yu, M. Winslett. An Unified Scheme for Resource Protection in Automated Trust Negotiation. In IEEE Symposium on Security and Privacy, May 2003