

# TECHNICAL RESEARCH REPORT

A Model for Access Negotiations in Dynamic Coalitions

*by Himanshu Khurana, Virgil D. Gligor*

**CSHCN TR 2003-16**  
**(ISR TR 2003-29)**



*The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.*

**Web site <http://www.isr.umd.edu/CSHCN/>**

# A Model for Access Negotiations in Dynamic Coalitions

Himanshu Khurana and Virgil D. Gligor

University of Maryland, College Park MD

{hkhurana, gligor}@eng.umd.edu

## Abstract

*The process of negotiating common access states in dynamic coalitions that comprise tens of autonomous domains sharing hundreds of resources is time-consuming and error-prone if performed without the benefit of automated tools. This process is also repetitive since, during the lifetime of a dynamic coalition, member domains must undertake the task of negotiating common access states multiple times as domains leave and new ones join the coalition. To define and verify the correctness of tools for automated negotiation, we develop a formal state-transition model of the process of negotiating a common access state. We extend an existing Role Based Access Control (RBAC) language to illustrate a wide variety of negotiation constraints and present a resolution procedure for verifying the satisfaction of such constraints in the state-transition model.*

*Keywords:* dynamic coalitions, access negotiation, state transition model, constraint language

*Technical Area:* system security, security models

## 1. Introduction

In various collaborative environments such as alliances for research and development, health care, airline route management, public emergency response, and military joint task forces, autonomous domains form coalitions to achieve common objectives. These coalitions can be dynamic in that member domains may leave or new domains may join after coalition establishment. Collaborative computing undertaken by dynamic coalitions requires a variety of resource-access agreements ranging from those for peer-to-peer sharing of applications and services to those for joint administration of access policies among the autonomous domains.

In dynamic coalitions, resource-access agreements ensure that all domains can have a common view of coalition operations and can execute shared applications and access shared objects. The result of such agreements is a *common access state*, which consists of the access permissions granted to the coalition users for executing shared applications/services, and to shared applications/services for accessing objects required by coalition operations. Reaching agreements among the domains of a coalition requires a negotiation process that is time-consuming and error-prone even for small-sized coalitions (i.e., coalitions consisting of five to ten autonomous domains), if performed without the benefit of automated tools.

Negotiation is time-consuming because the number of objects whose accessibility is being negotiated and the number of negotiation rounds may be large even if the number of applications is relatively small. For example, a route-sharing application among several airlines may include tens of route types, such as US-Asia, US-Europe and US-South America, and each route type may include tens of specific flights. Negotiating the access to each individual flight for hundreds of flights among five to ten airlines would require multiple rounds of flight-sharing proposals and proposal evaluations on the part of each airline, since each may have different goals for the negotiation outcome. It is possible that a negotiation round may not reach agreement, in which case a new round may have to be commenced for reaching the same goals. It is also possible that multiple common-access states may satisfy all the goals of the negotiating parties, in which case a common choice must be made.

The negotiation process is error-prone if performed without the benefit of automated tools, particularly when it is conducted under time constraints and when the size of a coalition increases to tens of domains sharing hundreds of resources (e.g., an international coalition of civilian and military organizations responding to international crises [22]). Furthermore, automated tools for common-access state negotiations in dynamic coalitions are required since the negotiation process is repetitive. That is,

during the lifetime of a coalition, member domains must undertake the task of negotiating common access states multiple times as domains leave and new ones join the coalition. Re-negotiation becomes necessary to add resources of joining domains, exclude departing domains from joint administration of coalition resources, and exclude resources withdrawn by departing domains.

To define and verify the correctness of automated tools for negotiating common access states in dynamic coalitions, we need to define the notion of access negotiation precisely. To this end, we develop a state-transition model of the process of negotiating a common access state, extend an existing Role Based Access Control (RBAC) language to illustrate a wide variety of negotiation constraints<sup>1</sup>, and present a resolution procedure for verifying the satisfaction of such constraints.

This paper is organized as follows. In Section 2 we classify negotiation constraints with examples and we motivate the notion of negotiating a common access state with an example. In Section 3 we motivate the need for a formal model of the negotiation process in the presence of coalition dynamics (i.e., domain join and leave events). In Section 4 we present the state-transition model for the process of negotiating common access states and a negotiation language for specifying constraints. In Section 5 we show that the state transition model supports the necessary security formulations. In Section 6 we discuss related work and we conclude in Section 7. In Appendix A we present elements and functions of our negotiation language. In Appendix B we present the semantics of the negotiation language. In Appendix C we provide proofs of theorems stated in Section 5 and in Appendix D we illustrate coalition resource negotiations in the state-transition model.

## 2. Negotiation of Common Access States: An Example

To achieve a common objective, the autonomous domains of a coalition negotiate the sharing of a set of resources (e.g., objects, applications, and services) and access permissions to those resources; i.e., they negotiate a common access state in which the coalition begins its operations. Negotiating a common access state means obtaining the agreement of each domain to share both privately owned and newly created resources, and to either privately or jointly administer [19] access to these resources. The negotiation result is not merely a union of the contributed resources necessary to achieve a coalition objective. Instead, the set of resources and their privileges contributed to the coalition by member domains must satisfy both *resource* and *permission constraints*. Examples of different types and sub-types of constraints are given in Table 1 below. Typically, these constraints arise from coalition objectives, access policies that are either jointly or privately enforced by autonomous domains, and resource-access requirements of coalition applications [10]. Further, these constraints can be either *global*, in which case they are known by all member domains, or *local*, in which case some constraints may remain private to some member domains. In either case, the specification of negotiation constraints is an important part of any access-policy specification and drives the negotiation process; e.g., it determines the number of negotiation rounds and the convergence to and commitment of common access states. Hence, it must be defined in a precise manner. For this reason, we define a negotiation language to specify constraints and present a resolution procedure for verifying the satisfaction of these constraints (viz., Section 4).

*An Example:* To illustrate the process of negotiating a common access state, assume that three domains representing three airlines that wish to share six types of routes form a coalition to expand their market coverage. Here, each route type corresponds to a certain set of departure and destination pairs (airline routes) in a given region, for instance, in U.S. - Europe, U.S. - Middle East, U.S. - North Africa, U.S. - Southern Africa, U.S. - Asia, and U.S. - South America. Sharing an individual route of a certain type implies that the airline domain that owns the route grants access permissions required to execute the route applications (e.g., reservations, billing, advertising) for that route to users of a foreign airline domain. Furthermore, airline domains negotiate and impose a set of pricing policies on all shared routes; e.g., policies on travel packages comprising multiple airline routes, vacation packages, and frequent flyer miles. Airline domains agree to enforce these policies on their privately owned routes that are shared with

---

<sup>1</sup> In this paper we assume that all domains have a common interpretation of the RBAC policy model [9].

the coalition members. To ensure that all member domains adhere to these pricing policies, member domains may decide to jointly setup and administer an auditing application that has access to shared airline routes in each domain. Such applications are vital to the coalition (as per the contractual agreement in this case) and the domains would like to jointly own and administer them to ensure that they remain with the coalition even after the departure of a member domain and that there is consensus on their access policy specification/modification [19]. (Note that privately owned resources that are shared with coalition members, such as airline routes, can be withdrawn from the coalition by the owner domain and their access policies are unilaterally specified). In general, coalition applications will require auditing of sensitive operations and the auditing applications are likely to be jointly administered.

Resource-based Constraints		Permission-based Constraints	
<i>Sub-type</i>	<i>Example</i>	<i>Sub-type</i>	<i>Example</i>
<i>Least Privilege</i> (Least privilege principle of resource sharing)	From a set of common applications, choose one that requires access to least number of objects	<i>Obligation</i> (A requirement for the presence – “must”, or absence – “must not” of permissions)	At least one user in every domain must be able to perform audit operations on every jointly administered application
<i>Cost-based</i> (Profit of sharing foreign resources minus cost of sharing local resources)	A domain requires access to specific foreign resource in order to share any of its own resources	<i>Separation-of-duty/ Prohibition</i> [1, 11] (Limits distribution of access permissions)	No role can have permissions for all the applications of any jointly administered resource
<i>Obligation</i> (A requirement for the presence – “must”, or absence – “must not” of resources)	If a domain controls a unique route application, it must be shared	<i>Cardinality</i> (Numerical limitation)	No role that has permissions for jointly administered applications can have more than $n$ members (users)

Table 1: Examples of Negotiation Constraints

In Figure 1 below, we denote the route types controlled by each of the three airline domains before negotiation as circles within the perimeter of each domain. The number of routes of each type is indicated in a parenthesis within each circle that each application controls. The arrow from a domain to a route type denotes that the domain desires that route type as an outcome of the negotiation. Observe that some route types and routes may already be common to multiple domains; e.g., route types 1, 2, 4, and 5. In this example, the objective of the negotiation is to obtain a common access state consisting of six shared route types (1 ... 6) among the three airline domains  $D_1$ ,  $D_2$ , and  $D_3$  and to jointly administer an auditing application that has access to all the shared routes.

Let us assume that the following two global negotiation constraints have been agreed upon to satisfy coalition objectives:

- Domains that control unique route types must share them with other domain (i.e., an obligation constraint). As a consequence, Domain 1 must share route type 6 and Domain 2 must share route type 3.
- Sharing of route types must minimize the number of routes shared (least privilege constraint); i.e., if two or more domains are capable of sharing the same route type, then the one that comprises the lowest number of routes will be used.

On inspection of Figure 1, we observe that there are two ways for domains to share their airline routes both of which satisfy the negotiation constraints; i.e., Domains  $D_1$ ,  $D_2$  and  $D_3$  could share either route types  $\{1,6\}$ ,  $\{3\}$  and  $\{2,4,5\}$  or route types  $\{6\}$ ,  $\{1,3\}$  and  $\{2,4,5\}$ . In general, there can be multiple common access states that satisfy the negotiated constraints and a common choice must be made. The negotiation proceeds as follows. The domains join the coalition and specify the negotiation constraints.

They then contribute resources that they are willing to share, namely, those in Figure 1. One of the domains makes a proposal for resource sharing that includes either one of the two common access states and the auditing application. The other domains vote on the proposal based on their own local negotiation constraints, if any<sup>2</sup>. If a proposal receives unanimous votes, it is accepted; else, a new round of negotiation follows with other proposals. Once the common set of shared route types is agreed upon, the three airline domains specify the users and user permissions for the resources (e.g., via role-membership in an RBAC system), thereby completing the definition of the desired common access state. (In general, the negotiation proposals may include <resource, access permission> pairs, not just resources as illustrated in this example.) The common access state is then committed. Note that if the negotiation does not converge to a desired common access state after an agreed upon number of rounds, negotiations will cease. This would signify the need to re-define the coalition objectives.

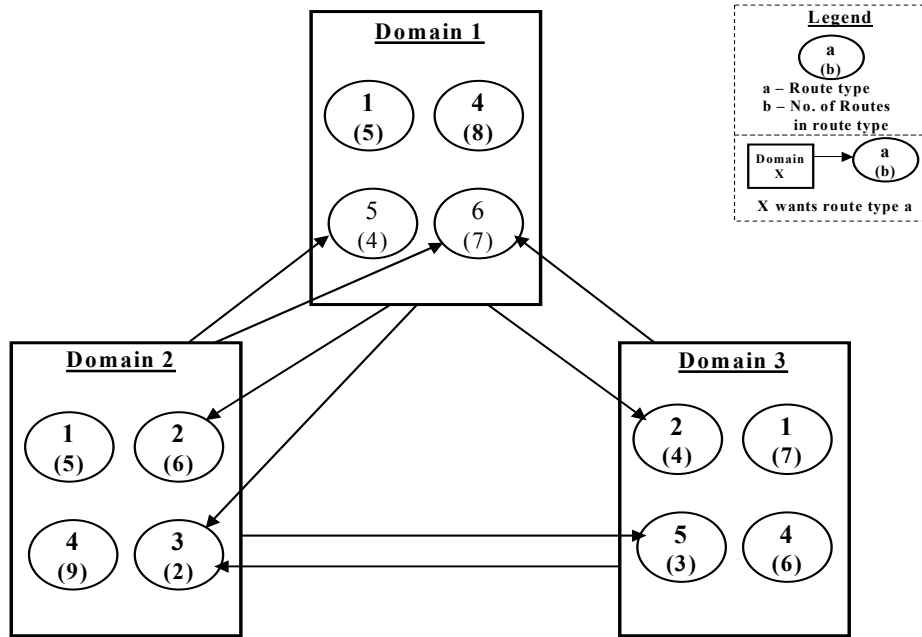


Figure 1: Routes controlled and desired before negotiation

### 3. Negotiations in the Presence of Coalition Dynamics

Dynamic coalitions have two characteristics that help motivate the key aspects of common access state negotiations, namely, membership that (1) comprises domains with overlapping but not identical interests that are managed under diverse access policies reflecting different sharing requirements, and (2) varies dynamically, thereby ruling out off-line, one-time negotiation of common access states. In dynamic coalitions, both domain departures and joins occur after coalition formation and require re-negotiation of the common access state. We consider the following three dynamic events and argue that each event leads to repeated negotiation of common access states.

- (a) *Domain-join event*: At any point in time after the coalition has been formed, member domains may agree to add a new domain to the coalition. This would be motivated by the coalition members' desire to share the joining domain's resources and willingness to share existing coalition resources with the joining domain to satisfy the joining domain's objectives. The sharing of new resources and changes in the sharing of existing resources requires re-negotiation of the common access state. For example,

<sup>2</sup> Examples of negotiations with local constraints are provided by Gligor *et al.* [12]

in the coalition of airline domains illustrated in Section 2, if a new airline domain joins the coalition, then the common access state would be re-negotiated among the four domains to include both the joining domain's airline routes and permissions and the joining domain's users and their access permissions to existing (shared) airline routes. Observe that all domain joins are entirely voluntary events in that both the joining domain and the existing coalition domains must agree to enable the join event to take place.

- (b) *Voluntary domain-departure event*: At any time after coalition formation, a domain can choose to leave the coalition and withdraw its (privately-owned) resources. However, the departing domain would not be able to withdraw any jointly owned and administered resources, if it was agreed prior to coalition formation that these resources were *vital* to the coalition's existence. In this case, the jointly owned resources would remain with the coalition after a domain departure. (This policy agreement can be enforced by *shared public-key mechanisms*, which inherently prohibit any single domain from withdrawing jointly owned resources [19]). Nonetheless, withdrawal of privately owned resources from the coalition can cause the remaining coalition domains to re-negotiate the common access state for the following two reasons.
- *Continued joint administration of resources*: After the departure of a domain, the remaining coalition domains must continue to perform joint administration of resources. This requires that they re-negotiate the policies of joint administration to exclude the departing domain and revoke any permissions previously distributed to the departing domain's users. This would cause re-negotiation of the common access state. For example, in the coalition of airline domains illustrated in Section 2, if one of the domains leaves, then the remaining domains would have to re-negotiate the access policies of the jointly owned auditing application to exclude the departing domain and to access only the shared routes of the domains remaining in the coalition.
  - *Withdrawal of resources essential to the coalition mission*: Even if the departing domain cannot withdraw jointly owned resources that are vital to the coalition's existence, the privately owned resources it withdraws may be *essential* to the coalition objectives. For example, in the coalition of airline domains, the departure of a domain may result in withdrawal of a route type that was essential to the coalition mission (e.g., route type 3, which is a unique route type owned by Domain 2). In this case, the remaining coalition domains would have to redefine the coalition objectives and re-negotiate the common access state to include resources that satisfy the redefined objectives. Furthermore, the withdrawal of essential resources may also result in the violation of negotiation constraints that were defined to ensure satisfaction of the coalition objectives (though they were satisfied prior to domain departure). For example, in the coalition of airline domains in the example presented in Section 2, if a negotiation constraint stated that there must be at least two unique route types in the common access state, then this constraint would be violated by the departure of Domain 2. Consequently, the common access state must be re-negotiated such that it either satisfies the existing constraints or re-defines them based on the redefined coalition objectives.
- (c) *Involuntary domain-departure event*: At any time after coalition formulation, it is possible that a majority of domains wish to exclude a domain from the coalition hereby causing the involuntary departure of that domain. Since we are primarily interested in cooperative coalitions, such an event can be viewed as a formation of a new coalition that excludes the departing domain. That is, the remaining domains would establish a new common access state rather than re-negotiating an existing one. However, domains may impose administrative policies on jointly owned resources that require re-negotiating the common access state after an involuntary departure-event. For example, such a policy might state that any domain that must leave involuntarily can withdraw the resources it contributed to the coalition and this policy may be part of the contractual agreement established by all member domains. Therefore, to satisfy such a contractual agreement, the remaining member domains must re-negotiate the common access state, which excludes the jointly owned resources contributed by the departing domain.

#### 4. The State Transition Model and the Negotiation Language

In this section we present our state transition model for negotiating a common access state and a negotiation language, NL, for expressing constraints of the negotiation. We present the model elements, the state variables, and the state transition functions. The model supports the following policy of resource-negotiation systems:

*Resource-Negotiation Policy:* A set of domains that comprise a coalition may share a set of privately owned resources and jointly own and administer resources specified in a common access state, only if (1) the domains contribute towards the common access state either their own resources or resources for joint ownership that are supported by well-defined access policies (i.e., permissions and role-to-permissions relations for the resources), (2) the domains unanimously agree on the common access state, (3) the common access state satisfies a specified set of local and global negotiation constraints, and (4) the common access state satisfies each domain’s specified local access constraints (e.g., separation-of-duty constraints).

*Assumptions for the state-transition model*

- ⚡ We assume that all domains have a common interpretation of the RBAC policy model.
- ⚡ In our model, we focus on cooperative coalitions and assume that domains do not lie about the resources they own or about the resource access policies they administer.

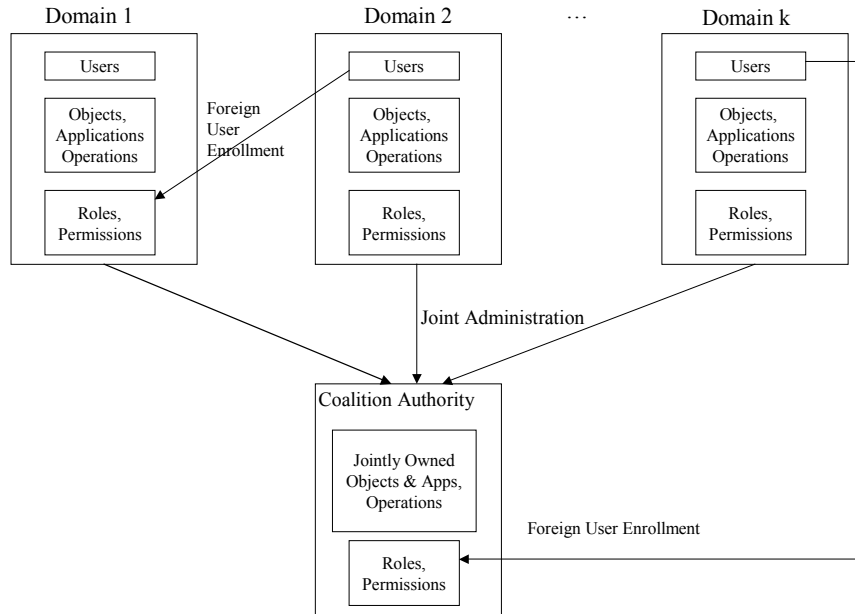


Figure 2: Model Elements

The resource-negotiation policy of the state-transition model specifies how the negotiation should proceed to allow coalition domains to share resources by satisfying negotiation constraints and ensuring that the access constraints of any member domain<sup>3</sup> are not violated. We define a secure state in the state-transition model in terms of a security invariant, which is a formal instantiation of the resource-negotiation policy. We present the model elements in Section 4.1, the negotiation language NL in Section 4.2, the state variables in Section 4.3, the security invariant and state transition rules in Section 4.4, and satisfaction of constraints in Section 4.5.

<sup>3</sup> Our goal is to model the process of negotiating a common access state. We do not present solutions for composing proposals for common access states that are likely to be accepted by all domains. Game theory or heuristics can provide such solutions; an introduction to game theoretic solutions is provided in [12].

#### 4.1. The Model Elements

The elements of the model are illustrated in Figure 2 above. Each domain has sets of users, roles, user-to-role assignment relations, objects, applications, operations, permissions (allowed operations on specific objects or applications), permission-to-role assignments relations and local access constraints. The access constraints are defined in the negotiation language NL and include permission-based constraints applied to the domain's users. We define an application a set of permissions; i.e., a set of operations allowed on a set of objects. (It may be useful to define a sequence of execution of permissions on objects for each application as discussed by Gligor and Gavrilu [10] but we ignore this *execution plan* for simplicity). We also do not support role hierarchies in our model for simplicity. For jointly owned resources we have elements similar to those for each domain's local resources. For negotiating resources, the elements include sets of domain proposals, votes on domain proposals, roles for access to negotiated resources, users that will access negotiated resources, and local and global negotiation constraints defined in NL. The list of elements is given below in Table 2.

#### 4.2. The Negotiation Language NL

The negotiation language NL is designed to specify (1) negotiation constraints (both local and global) and (2) domain access constraints. NL is an extension of RCL2000 [1]. We extend RCL2000 to include elements for domains comprising a coalition, jointly owned resources, assignment of foreign domain users to roles, functions on coalition-wide elements, and mathematical sets and functions useful for defining negotiation constraints. We discuss the syntax of NL here and the semantics in Appendix B.

<p>≠ <b>Domain user set <math>D_nU</math></b>: The set of all users in a domain <math>D_n</math> is denoted by <math>D_nU = \{user_1, user_2, \dots, user_z\}</math></p> <p>≠ <b>Domain role set <math>D_nR</math></b>: The set of all roles in a domain <math>D_n</math> is denoted by <math>D_nR = \{role_1, role_2, \dots, role_w\}</math></p> <p>≠ <b>Domain user-to-role assignment relation <math>D_nUA \geq D_nU \times D_nR</math></b> is a many-to-many user-to-role assignment relation for domain <math>D_n</math>.</p> <p>≠ <b>Domain foreign user-to-role assignment relation <math>D_nFUA \geq \{D_1U, D_2U, \dots, D_{n-1}U, D_{n+1}U, \dots, D_kU\} \times D_nR</math></b> is a many-to-many user-to-role assignment relation for users from all domains 1..k except domain <math>D_n</math> and roles in domain <math>D_n</math>.</p> <p>≠ <b>Domain object set <math>D_nOBJ</math></b>: The set of all objects in a domain <math>D_n</math> is denoted by <math>D_nOBJ = \{obj_1, obj_2, \dots, obj_m\}</math></p> <p>≠ <b>Domain operation set <math>D_nOP</math></b>: The set of all operations allowed in a domain <math>D_n</math> is denoted by <math>D_nOP = \{op_1, op_2, \dots, op_q\}</math></p> <p>≠ <b>Domain permission set <math>D_nP</math></b>: The set of all permissions in a domain <math>D_n</math> is denote by <math>D_nP = D_nOP \times D_nOBJ</math></p> <p>≠ <b>Domain permission-to-role assignment relation <math>D_nPA \geq D_nP \times D_nR</math></b> is a many-to-many permission-to-role assignment relation for domain <math>D_n</math>.</p> <p>≠ <b>Domain application set <math>D_nApp</math></b>: The set of all applications in a domain <math>D_nApp</math> is denoted by <math>D_nApp = \{app_1, app_2, \dots, app_u\}</math> where <math>app_i</math> is a set of permissions <math>P \geq D_nP</math>.</p> <p>≠ <b>Domain local access constraint set <math>D_nAC</math></b>: The set of all local domain access constraints on local domain objects is denoted by <math>D_nAC = \{const_1, const_2, \dots, const_c\}</math> where each constraint is defined in the NL.</p> <p>≠ <b>Joint Resources role set <math>JR</math></b>: The set of roles for jointly owned resources is denoted by <math>JR = \{role_1, \dots, role_s\}</math></p> <p>≠ <b>Joint Resource user-to-role assignment relation <math>JUA \geq \{D_1U \cong D_2U \cong \dots \cong D_kU\} \times JR</math></b> is a many-to-many user-to-role assignment relation for users of all domains <math>D_1 \dots D_k</math>.</p> <p>≠ <b>Jointly owned object set <math>JOBJ</math></b>: The set of all jointly owned objects is denoted by <math>JOBJ = \{obj_1, obj_2, \dots, obj_j\}</math></p> <p>≠ <b>Joint resources operation set <math>JOP</math></b>: The set of all operations allowed for joint owned resources is denoted by <math>JOP = \{op_1, op_2, \dots, op_t\}</math></p> <p>≠ <b>Jointly owned resource permission set <math>JP = JOP \times JOBJ</math></b></p> <p>≠ <b>Joint resource permission-to-role assignment relation <math>JPA \geq JP \times JR</math></b> is a many-to-many permission-to-role assignment relation for jointly owned resources.</p> <p>≠ <b>Jointly owned application set <math>JApp</math></b>: The set of all jointly owned applications is denoted by <math>JApp = \{app_1, app_2, \dots, app_u\}</math> where <math>app_i</math> is a set of permissions <math>P \geq JP</math>.</p> <p>≠ <b>Domain local negotiation constraint set <math>D_nC</math></b>: The set of all local domain constraints on all coalition resources is denoted by <math>D_nC = \{const_1, const_2, \dots, const_d\}</math> where each constraint <math>const_i</math> is defined in NL.</p> <p>≠ <b>Global negotiation constraint set <math>GC</math></b>: The set of all global constraints that apply to all coalition resources is</p>
---



<p>denoted by <math>GC = \{gconst_1, gconst_2, \dots, gconst_e\}</math> where each constraint <math>gconst_i</math> is defined in NL.</p> <p>≠# <b>Domain Proposal set <math>D_n\text{Prop}</math></b>: The set of all coalition resources that a domain wishes to be included in the common access state is denoted by <math>D_n\text{Prop} = \{D_1O'', D_1App'', D_2O'', D_2App'', \dots, D_nO'', D_nApp'', JO'', JApp''\}</math> where <math>D_iO''</math> and <math>D_iApp''</math> are the sets of objects and applications in domain <math>D_i</math> and <math>JO''</math> and <math>JApp''</math> are the sets of jointly owned objects and applications that domain <math>D_n</math> wishes to be part of the coalition.</p> <p>≠# <b>Domain Proposal Vote set <math>D_n\text{PVote}</math></b>: The set of all votes for a particular domain proposal <math>D_n\text{Prop}</math> is denoted by <math>D_n\text{PVote} = \{D_1\_vote, D_2\_vote, \dots, D_k\_vote\}</math> where <math>D_i\_vote</math> is a Boolean with values (yes, no) signifying domain <math>D_i</math>'s vote on the proposal <math>D_n\text{Prop}</math>.</p> <p>≠# <b>Domain Role addition set <math>D_n\_add\_R</math></b>: The set of all roles that have permissions for all negotiated resources of domain <math>D_n</math> to be provided by domain <math>D_n</math>. <math>D_n\_add\_R = \{role_1, role_2, \dots, role_m\}</math> where <math>role_i \subset D_nR</math>.</p> <p>≠# <b>Joint resource Role addition set <math>J\_add\_R</math></b>: The set of all coalition roles that have permissions for negotiated jointly owned coalition resources. <math>J\_add\_R = \{role_1, role_2, \dots, role_w\}</math> where <math>role_i \subset JR</math>.</p> <p>≠# <b>Domain User-to-role addition relation <math>D_n\_add\_UA</math></b>: A many-to-many user-to-role assignment relation for users of domain <math>D_n</math> and for all roles <math>\subset \{D_1\_add\_R, D_2\_add\_R, \dots, D_k\_add\_R, J\_add\_R\}</math>.</p>
--

Table 2: Elements of the State Transition Mode

#### 4.2.1. Elements and Functions of NL

NL has the following five basic elements defined for privately and jointly owned coalition resources: (1) users, (2) roles, (3) objects and applications, (4) operations, and (5) permissions. These elements were defined in the previous section. We do not include the notion of a *session* that is typically supported for most RBAC systems [1, 11]. This is because the negotiation constraints identified in Section 2 do not need the concept of a session and, furthermore, it is not clear how any constraint, such as dynamic separation of duty, can be defined or enforced on elements active in multiple sessions across multiple domains. It is, however, possible to define session-based constraints for a single domain; we do not support that for simplicity and instead refer to the work of Ahn and Sandhu [1] and Gligor *et al.* [11] for such constraints.

The basic functions supported by NL are as follows. The function *users* gives the members of a particular role, *roles* gives the set of roles associated with a set of users and a set of permissions, *permissions* gives a set of permissions associated with a particular role, *operations* gives the set of operations of a set of roles for a given set of resources, *objects* gives the set of objects for a particular permission, and *applications* gives the set of applications for a set of permissions. These functions are defined on individual domain sets as well as on the coalition sets. NL has two non-deterministic functions, *oneelement* (OE) and *allother* (AO). The function  $OE(X)$  allows us to get one element  $x_i$ , from a given set  $X$ . The function  $AO(X)$  allows us to get a set by taking out one element from the set  $X$ . These two functions are related as for any set  $P$ ,  $\{OE(P) \equiv AO(P)\} = P$ , though neither is a deterministic function. These functions were introduced by Chen and Sandhu [5] and allow the expression of various constraints without the use of logic quantifiers.

Additional elements of NL include conflicting role sets, conflicting permission sets and conflicting user sets. These sets allow us to define access constraints and negotiation constraints without the use of explicit negation. For example, a set of conflicting roles comprises roles whose users and/or permissions are required to be mutually disjoint based on organizational policies, a set of conflicting permissions is a set that a particular user must not have irrespective of his role membership, and so on.

Additional functions supported by NL include mathematical functions useful for defining negotiation constraints. These functions can be defined on a set of natural numbers  $N$  and a set of lists  $L$  in addition to the basic elements. We define some useful functions in Appendix A, which are similar to functions that can be implemented in Prolog [27]. Tools for resource negotiations may implement other useful mathematical functions.

#### 4.2.2. Syntax of Statements in NL

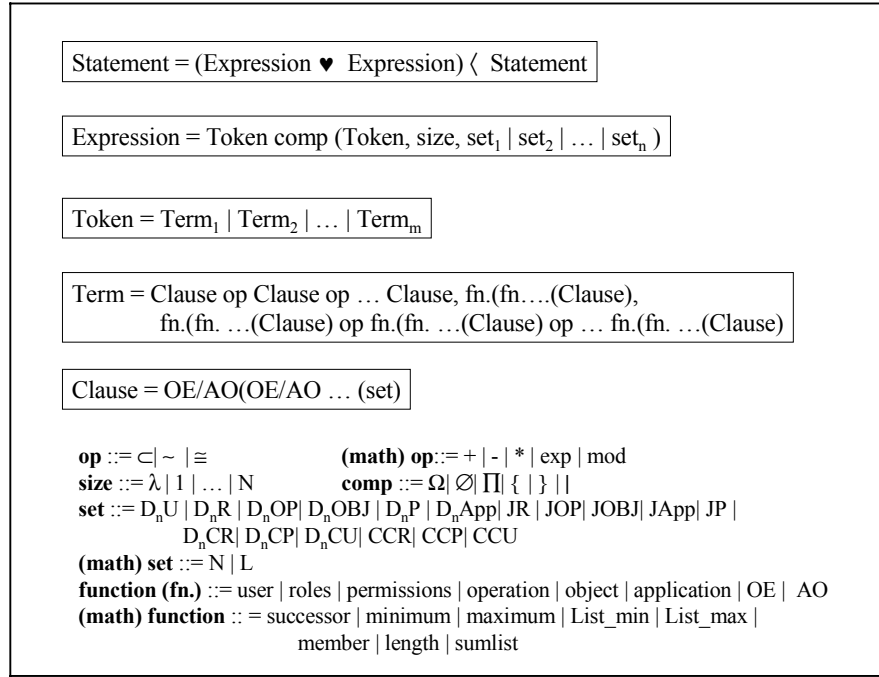


Figure 3: Syntax of NL Statements

In Figure 3 above we present the syntax of NL statements, which are formed from clauses of OE and AO functions defined on sets of NL. In Appendix B we show that in a manner similar to that for RCL2000 statements all NL statements have equivalent Restricted First Order Predicate Logic (RFOPL) statements. The syntax of RFOPL statements is similar to that of NL statements and is as follows.

- Every RFOPL statement has a (possibly empty) sequence of variables, *all* of which are universally quantified over sets in a left prefix to the statement
- The rest of the statement structure is similar to that in Figure 3 except for the structure of a clause, which is as follows: Clause = variable, set – variable

#### 4.3. State Model Variables

In the state-transition model, negotiations for resource sharing begin in a state where domain users have access to only their own local domain resources (i.e., the coalition does not exist). As the negotiations proceed, domains join the coalition, add resources to the coalition, propose common resource sharing, vote on proposals, and commit negotiated proposals. The state variables of the model record the state of the system at any given point in time in the form of information regarding the domain resources, proposals and commitments encountered in a coalition resource negotiation. These state variables are defined as follows in Table 3 below.

- |  |
|--|
| <p>≠# <b>Coalition Domain set CD</b>: At any given point in time, this set records all domains that are members of the coalition. The format of CD is: CD = {D<sub>1</sub>, D<sub>2</sub>, ...D<sub>k</sub>} where D<sub>i</sub> is a domain name</p> <p>≠# <b>Coalition Negotiation Constraint set CC</b>: At any given point in time, this set records all negotiation constraints (global and local) that are applied to coalition resources. The format of CC is: CC = {GC, D<sub>1</sub>C, D<sub>2</sub>C, ..., D<sub>k</sub>C } where entry GC is the set of all global constraints and entry (D<sub>i</sub>C) is a set of domain D<sub>i</sub>'s local negotiation constraints.</p> <p>≠# <b>Coalition Access Matrix CAM</b>: At any given point in time, this set records all (committed) users, roles, objects, applications, permissions, user-role mappings, permissions-role mappings, and local domain access</p> |
|--|

<p>constraints in all domains of the coalition including those for jointly owned resources. The format of CAM is: <math>CAM = \{(D_1U, D_1R, D_1OBJ, D_1App, D_1UA, D_1FUA, D_1OP, D_1P, D_1PA, D_1AC), (D_2U, D_2R, D_2OBJ, D_2App, D_2UA, D_2FUA, D_2OP, D_2P, D_2PA, D_2AC), \dots, (D_kU, D_kR, D_kOBJ, D_kApp, D_kUA, D_kFUA, D_kOP, D_kP, D_kPA, D_kAC), (JR, JOBJ, JApp, JOP, JP, JUA, JPA)\}</math> where entry <math>(D_iU, D_iR, D_iOBJ, D_iApp, D_iUA, D_iFUA, D_iOP, D_iP, D_iPA)</math> is the <i>domain access matrix</i>, <math>D_iAC</math> is the domain access constraints of domain <math>D_i</math>, and <math>(JR, JOBJ, JApp, JOP, JP, JUA, JPA)</math> is the access matrix for jointly owned resources.</p> <p>⊕ <b>Coalition Resource set CRes</b>: At any given point in time, this set records all objects and applications that coalition domains are willing to share with other = domains. The format of CRes is: <math>CRes = \{D_1O', D_1App', D_1Params, D_2O', D_2App', D_2Params, \dots, D_kO', D_kApp', D_kParams\}</math> where entry <math>D_iO' \geq D_iOBJ</math>, <math>D_iApp' \geq D_iApp</math> are the sets of all objects and applications that domain <math>D_i</math> is willing to share with other coalition domains and <math>D_iParams</math> are elements of NL used for verifying satisfaction of negotiation constraints.</p> <p>⊕ <b>Joint Resource set JRes</b>: At any given point in time, this set records all objects and applications that coalition domains would like to jointly own and administer with other coalition domains. The format of JRes is: <math>JRes = \{JOB, JApp, JR, JP, JOP, JPA, JParams\}</math> where JR, JOP, JP and JPA are the roles, operations, permissions, and role-permissions necessary to access jointly owned objects and applications in JOBJ and JApp, and JParams are elements of NL used for verifying satisfaction of negotiation constraints.</p> <p>⊕ <b>Coalition Proposal set CP</b>: At any given point in time, this set records the set of current domain proposals and all the votes on this domain proposal. The format of CP is: <math>CP = \{(D_1Prop, D_1PVote), (D_2Prop, D_2PVote), \dots, (D_kProp, D_kPVote)\}</math> where <math>D_iProp</math> is a proposal by domain <math>D_i</math> and <math>D_iPVote</math> is the set of all votes on that proposal.</p> <p>⊕ <b>Negotiated Resource set NCR</b>: At any given point in time, this set records the current negotiated set of coalition resources. The format of NCR is: <math>NCR = D_iProp \subset CP</math> for some domain <math>D_i</math>.</p> <p>⊕ <b>Coalition commit set CCOM</b>: At any given point in time, this set records all user-role memberships that coalition domains wish to add to the existing CAM. The format of CCOM is: <math>CCOM = \{D_1\_add\_R, D_1\_add\_UA, D_2\_add\_R, D_2\_add\_UA, \dots, D_k\_add\_R, D_k\_add\_UA, J\_add\_R\}</math>.</p> <p>⊕ <b>History set HIS</b>: At any given point in time, this set records the history of all state transitions on the state variables CD, CC, CAM, CRes, JRes, CP, NCR, and CCOM through multiple negotiations beginning with the formation of the coalition. The format of HIS is <math>HIS = \{history\_CD, history\_CC, history\_CAM, history\_CRes, history\_JRes, history\_CP, history\_NCR, history\_CCOM\}</math> where <math>history\_CRes</math> is a set of CRes elements through multiple negotiations, <math>history\_JRes</math> is a set of JRes elements through multiple negotiations, and so on.</p>
--

Table 3: State Model Variables

#### 4.4. The Secure State and the State Transition Rules

State (CD, CC, CAM, CRes, JRes, CP, NCR, CCOM, HIS): At any point in time, the state is defined as (CD, CC, CAM, CRes, JRes, CP, NCR, CCOM, HIS) where CD, CC, CAM, CRes, JRes, CP, NCR, CCOM and HIS are the state variables defined in Section 4.3.

We now define the secure state in the form of an invariant and the state transition rules supported by the model. The invariant is based on the resource-negotiation policy discussed at the beginning of Section 4. A state in the model includes the access policies and constraints of all coalition domains. It is likely that domains wish to keep their local policies and constraints private. Tools based on the state transition model can easily satisfy the privacy requirements of member domains by maintaining a distributed state across all domains with information regarding state transitions being communicated between the domains to maintain a secure state. In this work we assume that all elements of a state can be accessed at any given point in time and consequently the adherence to state transition rules can easily be verified.

*Secure State*: The definition of the secure state is a formulation of the negotiation in the form of a *State Invariant* SI, provided in Table 4 below, where a state (CD, CC, CAM, CRes, JRes, CP, NCR, CCOM, HIS) is said to be secure if:

<p><i>I1: <math>\&amp;X = (D_iU, D_iR, D_iOBJ, D_iApp, D_iUA, D_iFUA, D_iOP, D_iP, D_iPA, D_iAC) \subset CAM</math>:</i></p> <ul style="list-style-type: none"> <li>i) <math>D_i \subset CD</math></li> <li>ii) <math>D_iU \times D_iR \times D_iOBJ \times D_iApp \times D_iUA \times D_iFUA \times D_iOP \times D_iP \times D_iPA \models D_iAC</math></li> </ul> <p>This invariant states that all domains in the coalition must satisfy their local access constraints.</p> <p><i>I2: For <math>X = CC = \{GC, D_1C, D_2C, \dots, D_kC\}</math>  <math>\&amp;W = (gconst) \subset GC</math> and <math>\&amp;Y = (D_iC)</math>; <math>W, Y \subset X</math>,</i></p> <ul style="list-style-type: none"> <li>i) <math>W</math> is in NL and</li> <li>ii) <math>D_i \subset CD</math> and</li> <li>iii) <math>Y</math> is in NL and</li> <li>iv) <math>\&amp;Z = (D_iU, D_iR, D_iOBJ, D_iApp, D_iUA, D_iFUA, D_iOP, D_iP, D_iPA) \subset CAM: Z \models W \langle Z \models Y</math></li> </ul> <p>This invariant states that all domain access matrices must satisfy all negotiation constraints.</p> <p><i>I3: For <math>X = CRes = \{D_1O', D_1App', D_1Params, \dots, D_kO', D_kApp', D_kParams\}</math>  <math>\&amp;W = (D_iO') \subset X</math> and <math>\&amp;Y = (D_iApp') \subset X</math></i></p> <ul style="list-style-type: none"> <li>i) <math>D_i \subset CD</math> and</li> <li>ii) <math>W \geq D_iO \subset CAM</math> and</li> <li>iii) <math>Y \geq D_iApp \subset CAM</math></li> <li>iv) <math>D_iParams \subset X</math> are in NL</li> </ul> <p>This invariant states that all contributed coalition resources must be part of the domain access matrices.</p> <p><i>I4: For <math>X = JRes = \{JOB, JApp, JR, JP, JOP, JPA, JParams\}</math>  <math>\&amp;W = (obj) \subset JOB</math> and <math>\&amp;Y = (app) \subset Japp</math></i></p> <ul style="list-style-type: none"> <li>i) <math>( )</math> role <math>\subset JR</math>: <math>permissions(role) \geq JP \langle objects(permissions(role)) \neq W</math> and</li> <li>ii) <math>( )</math> role <math>\subset JR</math>: <math>permissions(role) \geq JP \langle applications(permissions(role)) \neq Y</math> and</li> <li>iii) <math>JParams \subset X</math> are in NL</li> </ul> <p>This invariant states that all jointly owned resources must have necessary roles and permissions associated with them.</p> <p><i>I5: For <math>X = CP = \{(D_1Prop, D_1PVvote), (D_2Prop, D_2PVvote), \dots, (D_kProp, D_kPVvote)\}</math></i></p> <ul style="list-style-type: none"> <li>i) <math>\&amp;Z = (D_jO'', D_jApp'') \subset X</math>: <math>Z \geq CRes</math></li> <li>ii) <math>\&amp;W = (JO'', JApp'') \subset X</math>: <math>W \geq Jres</math></li> <li>iii) <math>\&amp;(D_iProp, D_iPvvote)</math> in <math>X</math>: <math>D_iProp \models D_iC \equiv GC</math> where <math>D_iC \subset CC</math>, <math>GC \subset CC</math></li> </ul>	<p>This invariant states that each domain's proposal must include resources only from CRes and JRes and must satisfy the local domain negotiation constraints and the global negotiation constraints.</p> <p><i>I6: For <math>X = CP = \{(D_1Prop, D_1PVvote), (D_2Prop, D_2PVvote), \dots, (D_kProp, D_kPVvote)\}</math>  <math>\&amp;W = D_iPvote</math>, <math>\&amp;U = (D_jvote) \subset W</math>: <math> U  \Omega \langle U = yes</math> or <math>no</math></i></p> <p>This invariant states that each domain's proposal must be voted on by each of the coalition domains at most once with a "yes" or a "no" vote.</p> <p><i>I7: For <math>X = (D_iProp, D_iPVote) \subset NCR</math></i></p> <ul style="list-style-type: none"> <li>i) <math>D_iProp \subset CP</math> and</li> <li>ii) <math> D_iPVote  =  CD  \rangle Majority</math> and</li> <li>iii) <math>\&amp;Z = (D_i, vote) \subset D_iPvote</math>: <math>Z = yes</math></li> </ul> <p>This invariant states that the negotiated set of resources (NCR) is a proposal in CP that receives either unanimous votes from all coalition domains, or a pre-defined majority votes.</p> <p><i>I8: For <math>X = CCOM = \{D_1\_add\_R, D_1\_add\_UA, \dots, D_k\_add\_R, D_k\_add\_UA, J\_add\_R\}</math>  <math>\&amp;W = (D_i\_add\_R)</math>, <math>Y = (J\_add\_R)</math>; <math>W, Y \subset CCOM</math>,</i></p> <ul style="list-style-type: none"> <li>i) <math>objects(permissions(W)) \neq D_iO'' \langle applications(permissions(W)) \neq D_iApp''</math>; <math>D_iO'' \subset NCR</math>, <math>D_iApp'' \subset NCR</math></li> <li>ii) <math>objects(permissions(Y)) \neq JO'' \langle applications(permissions(Y)) \neq JApp''</math>; <math>JO'' \subset NCR</math>, <math>JApp'' \subset NCR</math></li> </ul> <p>This invariant states that each domain must provide roles with permissions for all of its resources in NCR and that the coalition authority must do the same for all jointly owned resources.</p> <p><i>I9: For <math>X = CCOM = \{D_1\_add\_R, D_1\_add\_UA, \dots, D_k\_add\_R, D_k\_add\_UA, J\_add\_R\}</math>  <math>\&amp;W = (D_i\_add\_R)</math>, <math>Y = (J\_add\_R)</math>, <math>\&amp;Z = (D_i\_add\_UA)</math>; <math>W, Y, Z \subset CCOM</math>,</i></p> <ul style="list-style-type: none"> <li>i) <math>\&amp;(u, v) \subset Z</math>: <math>u \subset D_iU</math>; <math>D_iU \subset CAM</math> and <math>v \subset \{W, Y\}</math></li> </ul> <p>This invariant states that all users that get access to the resources in NCR must be valid domain users and must get membership to only those roles specified in CCOM</p>
---	---

Table 4: State Invariant

*State transitions* occur when elements are added to the state variables as a result of resource negotiation taking place. Under the formulation of state as (CD, CC, CAM, CRes, JRes, CP, NCR, CCOM, HIS), resource negotiations can be completely represented as a sequence of state transition rules described below. We use the notation that the application of a state transition rule on a state  $S = (CD, CC,$

CAM, CRes, JRes, CP, NCR, CCOM, HIS) results in a new state denoted by  $S' = (CD', CC', CAM', CRes', JRes', CP', NCR', CCOM', HIS')$ . The state transition rules are provided in Table 5 and Table 6 below and examples of common access state negotiation using these rules are provided in Appendix D.

#### 4.5. Satisfying Constraints Specified in NL

In the state-transition model we specify state transition rules that require the satisfaction of a set of constraints. That is, given a set of statements (facts) the rule requires verification of the satisfaction of a given set of constraints against these facts. Since these constraints and facts in the coalition domains' RBAC models are specified in NL, we need a resolution procedure for NL that can be used for such verifications. As mentioned in Section 4.2, all statements in NL can be converted to RFOPL statements. All RFOPL statements are universally closed formulas with no negation and can therefore be represented as Horn Clauses, which are a subset of the syntax supported by Prolog [20, 27]. The process of verifying satisfaction of constraints required in our model is similar to that for verifying satisfaction of integrity constraints in deductive databases. Lloyd [20] defines a query process for verification of integrity constraints in deductive databases using the Prolog engine as a resolution procedure for the verification. Furthermore, Lloyd also shows that the Prolog engine is a sound resolution procedure and that the query process of verifying constraints is sound as well. Therefore, Prolog can also be used as a sound resolution procedure for verifying the satisfaction of constraints in our state-transition model (a subset of RCL2000 has been implemented in Prolog [23]). For development of languages for constraint programming we refer to Barth [3] and Comon *et al.* [6].

### 5. Security Formulations for the State Transition Model

In Section 4 we presented the state transition model for negotiating access to resources. We define the notion of a secure state in terms of security invariant SI and presented the state transition rules. We now present the security formulations for the model and show that the model supports these formulations.

*Initial State:* The initial state is defined to be  $(\{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\})$  and trivially satisfies the state invariant SI stated in Section 4.4.

The following two theorems show that the formulations of security based on the state invariant SI and the state transition rules are identical [21]. The proofs for these theorems are provided in Appendix C.

*Theorem 1:* If state  $S_n$  is the new state after application of a sequence of  $n$  state transition rules on state  $S_0$  and if  $S_0$  satisfies the invariant SI, then  $S_n$  also satisfies the invariant SI.

*Theorem 2:* If  $S_0 = (CD_0, CC_0, CAM_0, CRes_0, JRes_0, CP_0, NCR_0, CCOM_0, HIS_0)$  and  $S_n = (CD_n, CC_n, CAM_n, CRes_n, JRes_n, CP_n, NCR_n, CCOM_n, HIS_n)$  are two states that satisfy SI and  $CD_n \cong HIS_n \neq CD_0$ ,  $CC_n \cong HIS_n \neq CC_0$ ,  $CAM_n \cong HIS_n \neq CAM_0$ ,  $CRes_n \cong HIS_n \neq CRes_0$ ,  $JRes_n \cong HIS_n \neq JRes_0$ ,  $CP_n \cong HIS_n \neq CP_0$ ,  $NCR_n \cong HIS_n \neq NCR_0$ , and  $CCOM_n \cong HIS_n \neq CCOM_0$ , then there exists a sequence of rules that transforms  $S_0$  to  $S_n$  and is secure.

### 6. Related Work

*Negotiating Access to Coalition Resources.* Previous work in this area illustrates some of the important aspects of resource negotiations in specific settings. For example, Shands *et al.* [25] and Herzberg *et al.* [14] addressed the problem of unambiguously specifying a common access state, communicating this common state to all member domains, and committing this common access state. However, this work assumes that common access states were agreed upon by extra-technological (e.g., off-line) means and that all member domains have the same interpretation of the common policy model (i.e., a common interpretation of a role-based access control model). Other work addresses specific aspects of bilateral authorization-check negotiation, for instance those that enable clients and servers to agree on common authorization properties; i.e., matching client credentials with server access checks discussed by Seamons *et al.* [24] and Winsborough *et al.* [28]. Although this work on authorization-check negotiations introduces the important notion of client-server trust negotiation, it does not address the notion of common state negotiation in multiparty settings, such as those of dynamic coalitions.

*State Transition Rules*

<p><b>1) Rule 1: Join_coalition (CD, CAM, <math>D_i</math>, <math>\{D_iU, D_iR, D_iOBJ, D_iApp, D_iUA, D_iFUA, D_iOP, D_iP, D_iPA\}</math>)</b>  Semantics: When a domain joins the coalition, the domain's access matrix and access constraints are included in the coalition access matrix on the condition that the joining domain's access constraints are satisfied. Domains cannot join the coalition while a negotiation is in progress.  <b>The rule:</b>  <b>if</b> <math>\{D_iU \times D_iR \times D_iOBJ \times D_iApp \times D_iUA \times D_iFUA \times D_iOP \times D_iP \times D_iPA\} \models D_iAC</math> <b>and</b>  <math>CRes = JRes = CP = CCOM = \lambda</math>  <b>then</b> <math>CD' = CD \cong D_i</math>  <math>CAM' = CAM \cong \{D_iU, D_iR, D_iOBJ, D_iApp, D_iUA, D_iFUA, D_iOP, D_iP, D_iPA, D_iAC\}</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged</p> <p><b>2) Rule 2 : Define set of constraints (GC, <math>\{D_1C, D_2C, \dots, D_kC\}</math>)</b>  Semantics: The coalition members accept the set of constraints if the constraints can be satisfied by the CAM prior to resource negotiations. Constraints can be defined only at the beginning of a resource negotiation process.  <b>The rule:</b>  <b>if</b> <math>CAM \models GC \cong \{D_1C, D_2C, \dots, D_kC\}</math> <b>and</b>  <math>CRes = JRes = CP = CCOM = \lambda</math>  <b>then</b> <math>CC' = GC \cong \{D_1C, D_2C, \dots, D_kC\}</math>  all other state variable remain unchanged  <b>else</b> all state variables remain unchanged</p> <p><b>3) Rule 3 : Add domain resources for negotiation (CRes, <math>D_nOBJ'</math>, <math>D_nApp'</math>)</b>  Semantics: A domain may be willing to share a subset of its resources (objects and applications) based on its local preference and/or extra-technological agreements between coalition members.  <b>The rule:</b>  <b>if</b> <math>NCR = \lambda</math> <b>and</b>  <math>\&amp;X = (obj) \subset D_nOBJ': X \geq D_nOBJ \subset CAM</math>  <b>and</b> <math>\&amp;Y = (app) \subset D_nApp': Y \geq D_nApp \subset CAM</math>  <b>then</b> <math>CRes' = CRes \cong (D_nOBJ', D_nApp')</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged</p> <p><b>4) Rule 4 : Add resource for joint ownership (JRes, <math>\{JOBJ, JApp, JR, JOP, JP, JPA\}</math>)</b>  Semantics: A domain may wish to jointly own some resources. A domain may add these resources as long as it provides roles and permission relations to access the resources.</p>	<p><b>The rule:</b>  <b>if</b> <math>NCR = \lambda</math> <b>and</b>  <math>\&amp;X = (obj) \subset JOBJ: \text{role} \subset JR:</math>  <math>permissions(\text{role}) \geq JP \langle \text{objects}(permissions(\text{role})) \neq X</math>  <b>and</b> <math>\&amp;Y = (app) \subset JApp: \{role_1, role_2, \dots, role_r\} \subset JR: permissions(\text{role}) \geq JP \langle</math>  <math>applications(permissions(\text{role})) \neq Y</math>  <b>then</b> <math>JRes' = JRes \cong \{JOBJ, JApp, JR, JOP, JP, JPA\}</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged</p> <p><b>5) Rule 5 : Propose common shared resources (CP, <math>\{D_1O'', D_1App'', D_2O'', D_2App'', \dots, D_kO'', D_kApp'', JO'', JApp''\}</math>)</b>  Semantics: A domain <math>D_p</math> may propose any set of common shared resource that (1) consists of objects and applications from CRes and JRes and (2) satisfies domain <math>D_p</math>'s local negotiation constraints as well as the coalition global negotiation constraints.  <b>The rule:</b>  <b>if</b> <math>NCR = \lambda</math> <b>and</b>  <math>(D_1O'' \cong D_1App'' \cong D_2O'' \cong D_2App'' \cong \dots \cong D_kO'' \cong D_kApp'') \geq CRes</math> <b>and</b>  <math>JO'' \cong JApp'' \geq JRes</math> <b>and</b>  <math>\{D_1O'' \cong D_1App'' \cong \dots \cong D_kO'' \cong D_kApp'' \cong JO'' \cong JApp''\} \models \{D_pC \cong GC\}</math>  <b>then</b> <math>D_pProp = (D_1O'', D_1App'', \dots, D_kO'', D_kApp'', JO'', JApp'')</math>  <math>CP' = CP \cong D_pProp</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged</p> <p><b>6) Rule 6 : Vote for proposal (CP, <math>D_vProp</math>, <math>D_vPVote</math>, <math>D_vvote</math>)</b>  Semantics: A domain <math>D_v</math> may vote (yes, no) for any <math>D_iProp \subset CP</math>. A domain will vote "no" if its local constraints are not satisfied by the proposal.  <b>The rule:</b>  <b>if</b> <math>NCR = \lambda</math> <b>and</b>  <math>D_iProp \subset CP</math> <b>and</b>  <math>D_vvote \supset D_iPVote</math> <b>and</b>  <math>(D_iProp \models D_vC \cong GC \text{ and } D_vvote = \text{yes});</math>  <math>D_vC \subset CC, GC \subset CC</math>  <b>or</b> <math>(D_vvote = \text{no})</math>  <b>then</b> <math>D_iPVote' = D_iPVote \cong D_vvote</math>  <math>CP' = CP \cong D_iPVote'</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged</p>
--	---

Table 5: State Transition Rules 1 – 6

<p><b>7) Rule 7 : Declare negotiated set of shared resources (NCR)</b>  Semantics: The first proposal that receives either unanimous votes or a pre-defined <i>majority</i> votes is declared the negotiated common state  <b>The rule:</b>  <b>if</b> <math>\exists Y = D_iPVote \subset CP:  Y  =  CD  \rangle</math> <i>Majority</i> and  <math>\&amp;x = D_vvote \subset Y, x = \text{yes}</math>  <b>then</b> <math>NCR' = D_iProp \subset CP</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged  /* Note: negotiations may not end */</p> <p><b>8) Rule 8 : Add roles for committing NCR (CCOM, D<sub>i</sub>_add_R)</b>  Semantics: Each domain D<sub>i</sub> must provide a set of roles that have permissions for all its resources in NCR.  <b>The rule:</b>  <b>if</b> <math>D_i\_add\_R \geq D_iR \subset CAM</math> <b>and</b>  objects(permissions(D<sub>i</sub>_add_R)) <math>\neq D_iO' \subset NCR</math> <b>and</b>  applications(permissions(D<sub>i</sub>_add_R)) <math>\neq D_iApp' \subset NCR</math>  <b>then</b> <math>CCOM' = CCOM \cong D_i\_add\_R</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged</p> <p><b>9) Rule 9 : Add users for committing NCR (CCOM, D<sub>i</sub>_add_UA)</b>  Semantics: Each domain provides a set of users and a user-to-role relation mapping these users to roles in CCOM. The relation must satisfy the domain's local neg. and access constraints and global neg. constraints.  <b>The rule:</b>  <b>if</b> <math>\&amp;(x,y) \subset (D_i\_add\_UA): x \subset D_iU</math> and <math>y \subset \{D_{i1\_add\_R}, \dots, D_{ik\_add\_R}, J\_add\_R\}</math>  where <math>D_iU \subset CAM, \{D_{i1\_add\_R}, \dots, D_{ik\_add\_R}, J\_add\_R\} \subset CCOM</math> <b>and</b>  <math>D_i\_add\_UA \models D_iAC, D_iC \cong GC; D_iAC \subset CAM, D_iC, GC \subset CC</math>  <b>then</b> <math>CCOM' = CCOM \cong D_i\_add\_UA</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged</p> <p><b>10) Rule 10: Leave coalition (CAM, CC, D<sub>j</sub>, D<sub>j</sub>AM, HIS)</b>  Semantics: Any coalition domain may choose to leave the coalition at any time except when a negotiation is in progress. When a domain leaves, it withdraws its privately-owned resources. As a consequence any negotiation constraint (of the remaining coalition domains) that is no longer satisfied, is invalidated.</p>	<p><b>The rule:</b>  <b>if</b> <math>CRes = JRes = CP = NCR = CCOM = \lambda</math>  <b>then</b> <math>\&amp;(x,y) \subset D_jFUA: x \subset D_iU</math> for some <math>D_i \subset CD,</math>  <math>y \subset D_jR</math>  <math>CAM' = CAM - (x, y)</math>  <math>CC' = CC - D_jC</math>  <math>\&amp;const \subset CC': CAM' \sqcap const</math>  <math>CC' = CC' - const</math>  <math>CD' = CD - D_j</math>  <math>HIS' = HIS \cong CD \cong CC \cong CAM \cong CRes \cong JRes \cong CP \cong NCR \cong CCOM</math>  all other state variables remain the same  <b>else</b> all state variables remain unchanged</p> <p><b>11) Rule 11: Commit negotiated common state (CAM, HIS)</b>  Semantics: The negotiated NCR along with the roles, users, and user-to-role mappings in CCOM may be committed if they satisfy each domain's local access policy constraints and all negotiation constraints. All user-to-role mappings that were negotiated in previous negotiations but are not part of the current common access state must be revoked.  <b>The rule:</b>  <b>if</b> <math>NCR \cong CCOM \models CC \subset CAM</math> <b>and</b>  <math>NCR \cong CCOM \models \{D_1AC \cong D_2AC \cong \dots \cong D_kAC\} \subset CAM</math>  <b>then</b> <math>JResource = \{JOB, JApp, JR, JOP, JP, JPA\}</math>  {where <math>JOB \subset NCR, JApp \subset NCR, JR = J\_add\_R \subset CCOM, JP = \text{permissions}(JR), JOP = \text{operations}(JR, JOB, JApp)\}</math>  <math>CAM' = CAM \cong JResource</math>  /* add jointly owned resources to CAM' */  <math>\&amp;(x,y) \subset CCOM: x \subset D_iU, y \subset \{D_jR\}</math> for some <math>D_j \subset CD \langle (x,y) \supset CAM'</math>  <math>D_j\_FUA' = D_j\_FUA \cong (x, y)</math>  <math>CAM' = CAM' \cong D_j\_FUA'</math>  /* add foreign user-to-role relations */  <math>\&amp;(x,y) \subset CCOM: x \subset D_iU, y \subset JR</math> and <math>(x,y) \supset CAM'</math>  <math>JUA' = JUA \cong (x, y)</math>  <math>CAM' = CAM' \cong JUA'</math>  /* add joint admin. user-to- role relations to CAM' */  <math>\&amp;D_i \subset CD, \&amp;(x,y) \subset D_iFUA \subset CAM'</math> and <math>(x,y) \supset CCOM</math>  <math>CAM' = CAM' - (x,y) \setminus</math>  /* revoke prior user-to-role relations */  <math>\&amp;(x,y) \subset JUA \subset CAM'</math> and <math>(x,y) \supset CCOM</math>  <math>CAM' = CAM' - (x,y)</math>  /* revoke prior user-to-role relations */  <math>HIS' = HIS \cong CD \cong CC \cong CAM \cong CRes \cong JRes \cong CP \cong NCR \cong CCOM</math>  <math>CRes' = JRes' = CP' = NCR' = CCOM' = \lambda</math>  all other state variables remain unchanged  <b>else</b> all state variables remain unchanged</p>
---	--

Table 6: State Transition Rules 7 – 11

In contrast, our work explores the automation of the process of negotiating a common access state in a generic dynamic coalition multiparty setting where all the domains of a coalition share the same interpretation of a common policy model. We cast this negotiation problem as one of satisfying (1) diverse coalition-member objectives, (2) a specified set of negotiation constraints, and (3) the existing access constraints of the coalition members.

*A Language for Specifying Negotiation Constraints.* For negotiation a common access state, we require a language that is *expressive* in that it can be used to express a variety of negotiation constraints, and *easy to use* in that using the language should not require knowledge of underlying logic structures and rules. This is because we envision the language to be used by coalition administrators and not by security access policy designers. Furthermore, we require the language to be supported by a *resolution procedure* for verifying the satisfaction of negotiation constraints.

Tower [15] is an expressive language that is designed for object-oriented systems. It is easy to use for systems designers who understand object-oriented systems. However, the language allows for the definition of only a limited variety of constraints and there is no resolution procedure currently defined or implemented. The Ponder [7] policy specification language provides a common means for specifying security policies that map onto various access control implementation mechanisms. The language is targeted for management of large-scale object oriented systems and consequently the use of the language requires familiarity with these systems. A deployment model for object-oriented systems is discussed in [8], however, the language can express only a limited set of constraints. ASL [16, 17] and the authorization constraint language proposed by Bonatti *et al.* [4] are expressive languages for RBAC systems defined with well-formed logic structure and rules. The languages are supported by resolution procedures, which are shown to be implementable. Furthermore, the languages allows for the expression of conflicting authorizations that can be resolved by user-defined resolution mechanisms (functions). However, both the languages are difficult to use, as their use requires knowledge of underlying logic structures and rules. OASIS [13] is an RBAC architecture for interoperation of services in distributed environments. It uses the notion of role activation based on rules that are dependant on credentials, which are transferred via role membership certificates. OASIS defines an RBAC language that can be used with some background knowledge of logic structures and rules. However, the language is not very expressive for constraint definition. RCL2000 [1] is an expressive RBAC language targeted for flexible RBAC constraint specification. It is easy to use and logic structures are hidden from the language user. The language has been modeled in UML [26] and OCL [2] for constraint specification, and a subset of the language has been implemented in Prolog [23]. We have extended RCL2000 with additional elements and functions and shown that Prolog can be used as a resolution procedure for the language.

## 7. Conclusions

We have shown that negotiating a common access state means obtaining the agreement of each domain to share privately owned resources and to either privately or jointly administer access to the resources. We have also shown that in the presence of coalition dynamics, member domains must re-negotiate the common access state multiple times. We developed a state transition model of the negotiation process and a negotiation language for the specification of a wide variety of negotiation constraints, and we illustrated negotiations of common access states.

Several problems of coalition resource negotiation related to our work remain to be solved. Though we identify useful types of negotiation constraints, we do not identify all possible types. The task of formal characterization of negotiation constraints still remains to be undertaken. We assume that all domains have a common interpretation of the RBAC policy model. Future work on formal modeling of the negotiation process can relax this restriction and allow the negotiation of common access states that can be committed on multiple access policy models (e.g., by negotiating directly on permissions of the resources as opposed to negotiating membership to roles that have permissions for the resources). A problem closely related to our work is that of composing proposals for resource sharing that are likely to be accepted by all coalition domains. Game theory or heuristics may provide solutions to this problem.



## References

1. G-J. Ahn and R. Sandhu, "Role-based Authorization Constraints Specification", *ACM Transactions on Information and System Security*, pages 207-226, Vol. 3, No. 4, ACM, November 2000.
2. G-J. Ahn and M. E. Shin, "Role-based Authorization Constraints Specification Using Object Constraint Language", In Proceedings of 6th IEEE International Workshop on Enterprise Security (WETICE 2001), MIT, MA, June 20-22, 2001.
3. P. Barth, "Logic-Based 0-1 Constraint Programming", *Kluwer Academic Publishers*, Massachusetts 1996.
4. P. Bonatti, S. De Capitani di Vimercati, P. Samarati, "An Algebra for Composing Access Control Policies", in *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, n. 1, February 2002, pp. 1-35.
5. F. Chen and R. Sandhu, "Constraints for Role Based Access Control", Proceedings of the first ACM Workshop on Role Based Access Control, MD, December 1995, pp.39-46.
6. H. Comon, C. Marché, R. Treinen (Eds.), "Constraints in Computational Logics: Theory and Applications", International Summer School, CCL'99 Gif-sur-Yvette, France, September 5-8, 1999, Revised Lectures, *Lecture Notes in Computer Science Springer* 2002.
7. N. Damianou, N. Dulay, E. Lupu, M Sloman, "The Ponder Specification Language", International Workshop on Policies for Distributed Systems and Networks (Policy2001), HP Labs Bristol, Jan 2001.
8. N. Dulay, E. Lupu, M Sloman, N. Damianou, "A Policy Deployment Model for the Ponder Language", Proceedings of the IEEE/IFIP International Symposium on Integrated Network Management, Seattle, May 2001.
9. D. Ferraiolo, J. Cugini, and D. R. Kuhn, "Role-based access control: Features and motivations", Proceedings of the Annual Computer Security Applications Conference, 1995.
10. V.D. Gligor and S.I. Gavrilă, Application-Oriented Security Policies and their Composition, Proceedings of Security Protocols 6th International Workshop. Cambridge, UK, April 1998.
11. V.D. Gligor, S.I. Gavrilă, and D. Ferraiolo, "On the Formal Definition of Separation-of-Duty Policies and their Composition", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, May 1998.
12. V. D. Gligor, H. Khurana, R. Koleva, V. Bharadwaj, and J. Baras, "On the Negotiation of Access Control Policies", To appear in the 9th Security Protocols Workshop, Cambridge, UK, Springer-Verlag, 2001.
13. R. Hayton, J. Bacon, K. Moody, "OASIS: Access Control in an Open, Distributed Environment", Proceedings of the IEEE Symposium on Security and Privacy, Oakland CA, pp3-14, May 1998.
14. A. Herzberg, Y. Mass, J. Michaeli, D. Naor and Y. Ravid, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers", Proceeding of the IEEE Symposium on Security and Privacy, Oakland, California, May 2000.
15. M. Hitchens, V. Varadharajan, "Tower: A Language for Role Based Access Control", International Workshop on Policies for Distributed Systems and Networks, POLICY 2001 Bristol, UK, January, 2001, Springer-Verlag LNCS1995, pp. 88-106, 2001.
16. S. Jajodia, P. Samarati, V. S. Subrahmanian, "A logical language for expressing authorizations", Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, May 1997, pages 31-42.
17. S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian, "Flexible Support for Multiple Access Control Policies", *ACM Transactions on Database Systems*, pp. 214-260, June 2001.
18. H.Khurana, "Negotiation and Management of Coalition Resources", PhD Dissertation, University of Maryland, College Park, MD, June 2002.
19. H.Khurana, V.D. Gligor, and J. Linn, "Reasoning About Joint Administration of Access Policies for Coalition Resources", IEEE International Conference on Distributed Computing Systems, Vienna, July 2002 (full length manuscript available at <http://glue.umd.edu/~gligor>).
20. J.W. Lloyd, "Foundations of Logic Programming", Second Edition, *Springer-Verlag New York* 1987.

21. J. McLean, "Reasoning about Security Models", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, pp. 123-131, April 1987.
22. C. E. Phillips, T. C. Ting, S. A. Demurjian, "Information sharing and security in dynamic coalitions" Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT), Moenterey, CA June 2002.
23. A. Schaad, "Detecting Conflicts in a Role-based Delegation Model", Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC), New Orleans, Louisiana, December 2001.
24. K. E. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation", Proceedings of the Internet Society's Symposium on Network and Distributed System Security, San Diego, CA, February 2001.
25. D. Shands, R. Yee, J. Jacobs, "Secure Virtual Enclaves: Supporting Coalition Use of Distributed Application Technologies", Proceedings of the Network and Distributed Systems Security Symposium, San Diego, February 2000.
26. M. E. Shin and G-J. Ahn, "UML-based Representation of Role-based Access Control", Proceedings of 5th IEEE International Workshop on Enterprise Security (WETICE 2000), NIST, MD, June 14-16, 2000.
27. L. Sterling, E. Shapiro, "The Art of Prolog", Second Edition, *The MIT Press*, Massachusetts, 1994.
28. W. H. Winsborough, K. E. Seamons, and V. E. Jones, "Automated Trust Negotiation", DARPA Information Survivability Conference and Exposition, Hilton Head, January 2000.

## Appendix A: Elements and Functions of Negotiation Language NL

### **Basic Elements and functions**

- Model Elements, namely,  $D_iU$ ,  $D_iR$ ,  $D_iOBJ$ ,  $D_iApp$ ,  $D_iUA$ ,  $D_iFUA$ ,  $D_iOP$ ,  $D_iP$ ,  $D_iPA$ ,  $D_iAC$ ,  $JR$ ,  $JOBJ$ ,  $JApp$ ,  $JOP$ ,  $JP$ ,  $JUA$ ,  $JP$
- users :  $D_nR \Downarrow 2^U$ , a function mapping each domain  $D_n$  role  $r_i$  to a set of users
- roles:  $D_nU \cong D_nP \Downarrow 2^R$ , a function mapping the set  $D_nU$  and  $D_nP$  to a set of roles  $R$
- permissions:  $D_nR \Downarrow 2^P$ , a function mapping each domain  $D_n$  role  $r_i$  to a set of permissions
- operations:  $D_nR \times (D_nOBJ, D_nApp) \Downarrow 2^{OP}$ , a function mapping each domain  $D_n$  role  $r_i$ , and object/application  $obj_i/app_i$ , to a set of operations
  - o operations  $(r_i, obj_i) = \{op \in D_nOP \mid (op, obj_i, r_i) \in D_nPA\}$
- objects:  $D_nP \Downarrow 2^{OBJ}$ , a function mapping each domain  $D_n$  permission  $p_i$  to a set of objects
- applications:  $D_nP \Downarrow 2^{App}$ , a function mapping a set of domain  $D_n$  permissions to a set of domain  $D_n$  applications
- coalition users:  $\{D_1R, \dots, D_nR, JR\} \Downarrow 2^U$ , a function mapping each coalition role  $r_i$  to a set of coalition users
- coalition roles:  $\{D_1U, \dots, D_nU\} \cong \{D_1P, \dots, D_nP, JP\} \Downarrow 2^R$ , a function mapping the set of coalition users and coalition permissions to a set of coalition roles
- coalition permissions:  $\{D_1R, \dots, D_nR, JR\} \Downarrow 2^P$ , a function mapping each coalition role  $r_i$  to a set of permissions
- coalition operations:  $\{D_1R, \dots, D_nR, JR\} \times \{D_1OBJ, D_1App, \dots, D_nOBJ, D_nApp, JOBJ\} \Downarrow 2^{OP}$ , a function mapping each coalition role  $r_i$ , and coalition object/application  $obj_i$ , to a set of operations
  - o coalition operations  $(r_i, obj_i) = \{op \in \{D_1OP, \dots, D_nOP, JOP\} \mid (op, obj_i, r_i) \in \{D_1PA, \dots, D_nPA, JPA\}\}$
- coalition objects:  $\{D_1P, \dots, D_nP, JP\} \Downarrow 2^{OBJ}$ , a function mapping each coalition permission  $p_i$  to a set of objects
- coalition applications:  $\{D_1P, \dots, D_nP, JP\} \Downarrow 2^{App}$ , a function mapping a set of coalition permissions to a set of applications
- $D_nCR$  = a collection of domain conflicting role sets,  $\{cr_1, \dots, cr_s\}$ , where  $cr_i = \{r_1, \dots, r_t\} \geq D_nR$
- $D_nCP$  = a collection of domain conflicting permission sets,  $\{cp_1, \dots, cp_u\}$ , where  $cp_i = \{p_1, \dots, p_v\} \geq D_nP$
- $D_nCU$  = a collection of domain conflicting user sets,  $\{cu_1, \dots, cu_w\}$ , where  $cu_i = \{u_1, \dots, u_x\} \geq D_nU$

- $D_n\text{COBJ}$  = a collection of domain conflicting object sets,  $\{\text{cobj}_1, \dots, \text{cobj}_w\}$ , where  $\text{cobj}_i = \{\text{obj}_1, \dots, \text{obj}_x\} \geq D_n\text{OBJ}$
- $D_n\text{CApp}$  = a collection of domain conflicting application sets,  $\{\text{capp}_1, \dots, \text{capp}_w\}$ , where  $\text{capp}_i = \{\text{app}_1, \dots, \text{app}_x\} \geq D_n\text{App}$
- $\text{CCR}$  = a collection of coalition conflicting role sets,  $\{\text{cr}_1, \dots, \text{cr}_s\}$ , where  $\text{cr}_i = \{r_1, \dots, r_t\} \geq \{D_1R \equiv D_2R \equiv \dots \equiv D_kR \equiv JR\}$
- $\text{CCP}$  = a collection of coalition conflicting permission sets,  $\{\text{cp}_1, \dots, \text{cp}_u\}$ , where  $\text{cp}_i = \{p_1, \dots, p_v\} \geq \{D_1P \equiv D_2P \equiv \dots \equiv D_kP \equiv JP\}$
- $\text{CCU}$  = a collection of coalition conflicting user sets,  $\{\text{cu}_1, \dots, \text{cu}_w\}$ , where  $\text{cu}_i = \{u_1, \dots, u_x\} \geq \{D_1U \equiv D_2U \equiv \dots \equiv D_kU\}$
- $\text{CCOBJ}$  = a collection of coalition conflicting object sets,  $\{\text{cobj}_1, \dots, \text{cobj}_w\}$ , where  $\text{cobj}_i = \{\text{obj}_1, \dots, \text{obj}_x\} \geq \{D_1\text{OBJ} \equiv D_2\text{OBJ} \equiv \dots \equiv D_k\text{OBJ} \equiv \text{JOBj}\}$
- $\text{CCApp}$  = a collection of coalition conflicting application sets,  $\{\text{capp}_1, \dots, \text{capp}_w\}$ , where  $\text{capp}_i = \{\text{app}_1, \dots, \text{app}_x\} \geq \{D_1\text{App} \equiv D_2\text{App} \equiv \dots \equiv D_k\text{App} \equiv \text{JApp}\}$
- $\text{onelement}(X) = x_i$ , where  $x_i \subset X$
- $\text{allother}(X) = X - \{\text{OE}(X)\}$

### **Additional Elements and Functions of NL**

- $N$ : the set of natural numbers
- $L$ : a set of lists where each list  $l = []$ . if empty, or  $l = [X|Y]$  with  $X$  = head of  $l$  and  $Y$  is another list and the tail of  $l$  where  $X$  can be a basic element of NL or a natural number
- Successor:  $N \Downarrow N$ , a function mapping a natural number to its successor, that is,  $\text{Successor}(n) = n + 1$  where  $n \in N$
- Minimum:  $N \times N \Downarrow N$ , a function mapping two natural numbers to the one that is the minimum, that is,  $\text{Minimum}(n, m, n) \Uparrow n \Omega m$ , where “ $\Uparrow$ ” denotes implication
- Maximum:  $N \times N \Downarrow N$ , a function mapping two natural numbers to the one that is the maximum, that is,  $\text{Maximum}(n, m, n) \Uparrow n \Omega m$
- Sumlist:  $L \Downarrow N$ , a function mapping a list of natural numbers to the number that is the sum of all the elements of the list, recursively defined as follows:
  - o  $\text{Sumlist}([], 0)$  /\* the sum of an empty list is 0 \*/
  - o  $\text{Sumlist}([X|Xs], \text{Sum}) \Uparrow \text{Sumlist}(Xs, \text{Issum})$  /\* Sum is  $X + \text{Issum}$  \*/
- List\_min:  $L \Downarrow N$ , a function mapping a list of natural numbers to a natural number which is the minimum in the list, defined as follows:
  - o  $\text{List\_min}([X|Xs], M) \Uparrow \text{List\_min}(Xs, X, M)$
  - o  $\text{List\_min}([X|Xs], Y, M) \Uparrow \text{Minimum}(X, Y, Y1), \text{List\_min}(Xs, Y1, M)$
- List\_max:  $L \Downarrow N$ , a function mapping a list of natural numbers to a natural number which is the maximum in the list, that is,  $\text{Listmax}(Xs, N)$  defined similar to  $\text{List\_min}$ .
- Member:  $L \times L \Downarrow T/F$ , a function mapping a list  $l (l = [X|Y])$  and an element of a list  $x$  to a Boolean value (true/false) as follows:
  - o  $\text{Member}(X, [X|Xs])$
  - o  $\text{Member}(X, [Y|Ys]) \Uparrow \text{Member}(X, Ys)$
- Length:  $L \Downarrow N$ , a function mapping a list  $l (l = [X|Y])$  to natural number  $n$  identifying the number of elements of the list and is defined as follows:
  - o  $\text{Length}([], 0)$  /\* length of an empty list is 0 \*/
  - o  $\text{Length}([X|Xs], N) \Uparrow \text{Length}(Xs, N1)$  /\*  $N$  is  $N+1$  \*/

## **Appendix B: Semantics of Negotiation Language NL**

We present a *reduction* algorithm to convert any statement in the negotiation language NL to a statement in a restricted first order predicate logic (RFOPL) language and a *construction* algorithm to convert an RFOPL statement to a NL statement. These algorithms are similar to those given for the conversions between RCL2000 statements and

RFOPL statements [1]. We show that the relationship between the statements in NL and RFOPL is sound and complete where the proof for completeness is simpler than the one presented in [1].

### B.1. Conversions between NL and RFOPL Statements

The reduction of NL statements to RFOPL statements involves the elimination of OE and AO functions by replacing them with universal quantifiers while the construction of NL statements from RFOPL statements involves the elimination of quantifiers by replacing them with OE and AO functions. We present the reduction and construction algorithms below and note that the reduction algorithm is a modified version of the one presented in [1] for RCL2000 statements. This modification involves restricting the possible reductions of an NL statement to exactly one RFOPL statement and this modification allows us to simplify the completeness proof in Section B.2.

#### Reduction Algorithm

Input: NL statement; Output: RFOPL statement

Let Reduce-OE term be either  $OE(set)$  or  $OE(function(element))$ , where  $set$  is an element of  $\{D_iU, D_iR, D_iOBJ, D_iApp, D_iUA, D_iFUA, D_iOP, D_iP, D_iPA, D_iAC, JR, JOBJ, JApp, JOP, JP, JUA, JP\}$  and  $function$  is an element of  $\{user, roles, permissions, operations, object, application, coalition user, coalition roles, coalition permissions, coalition operations, coalition object, coalition application, minimum, maximum, list\_min, list\_max, member, sumlist, successor\}$

1. AO elimination

replace all occurrences of  $AO(expr)$  with  $(expr - \{OE(expr)\})$ ;

2. OE elimination

While there exists Reduce-OE term in NL statement

choose the innermost Reduce-OE term of the leftmost expression that has  
a Reduce-OE term

call *reduction* procedure;

End

Procedure *reduction*

case (i) Reduce-OE term is  $OE(set)$

create new variable  $x$ ;

put  $\&x \subset set$  to the right of the existing quantifier(s);

replace all occurrences of  $OE(set)$  by  $x$ ;

case (ii) Reduce-OE term is  $OE(function(element))$

create new variable  $x$ ;

put  $\&x \subset function(element)$  to the right of existing quantifier(s);

replace all occurrences of  $OE(function(element))$  by  $x$ ;

End

#### Construction Algorithm

Input: RFOPL statement; Output: NL statement

1. Construct NL statement from RFOPL statement

While there exists a quantifier in RFOPL statement

choose the rightmost quantifier  $\&x \subset X$ ;

pick values  $x$  and  $X$  from the chosen quantifier;

replace all occurrences of  $x$  by  $OE(X)$ ;

End

2. Replacement of AO

If there is  $(expr - \{OE(expr)\})$  in RFOPL statement

replace it with  $AO(expr)$ ;

### B.2. Relationship between NL Statements and RFOPL Statements

We have the following four lemmas based on the reduction ( $\mathcal{R}$ ) and construction ( $\mathcal{C}$ ) algorithms presented in the previous section.

*Lemma 1:* Given NL statement  $\zeta$ ,  $\mathcal{R}(\zeta)$  always gives the same RFOPL statement  $\eta$ .

*Proof:* The reduction algorithm always chooses the innermost OE function of the leftmost expression that has an OE function to reduce a NL statement to a RFOPL statement. This procedure is deterministic. Therefore, given NL statement  $\zeta$ , we will always get the same RFOPL statement  $\eta$ .

*Lemma 2:* Given RFOPL statement  $\eta$ ,  $C(\eta)$  always gives the same NL statement  $\zeta$ .

*Proof:* The construction algorithm always chooses the rightmost quantifier to construct a NL statement from the RFOPL statement. This procedure is deterministic. Therefore, given RFOPL statement  $\eta$ , we will always get the same NL statement  $\zeta$ .

*Definition: Intermediate statements* – The statements generated during reduction and construction are called intermediate statements. These statements have both quantifiers and OE terms.

*Lemma 3:* If the intermediate expression  $\tau$  is derived from an NL statement  $\zeta$  by the Reduction algorithm in  $k$  iterations, then the Construction algorithm applied to  $\tau$  will terminate in exactly  $k$  iterations.

*Proof:* Since the Reduction algorithm generates exactly one quantifier per iteration,  $\tau$  must have exactly  $k$  quantifiers. Furthermore, since the Construction algorithm eliminates exactly one quantifier per iteration, it will terminate in exactly  $k$  iterations.

*Lemma 4:* If the intermediate expression  $\tau$  is derived from an RFOPL statement  $\eta$  by the Construction algorithm in  $k$  iterations, then the Reduction algorithm applied to  $\tau$  will terminate in exactly  $k$  iterations.

*Proof:* Since the Construction algorithm generates exactly one OE term per iteration,  $\tau$  must have exactly  $k$  OE terms. Furthermore, since the Reduction algorithm eliminates exactly one OE-term per iteration, it will terminate in exactly  $k$  iterations.

We now prove the soundness and completeness of the relationship between NL and RFOPL statements.

*Soundness theorem:* Given NL statement  $\zeta$ ,  $\zeta$  can be translated into an RFOPL statement  $\eta$ . Also  $\zeta$  can be reconstructed from  $\eta$ . That is  $C(R(\zeta)) = \zeta$ .

*Proof:* Let us define  $C^n$  as  $n$  iterations of the Construction algorithm and  $R^n$  as  $n$  iterations of the Reduction algorithm. We show that  $C^n(R^n(\zeta)) = \zeta$  by induction on the number of iterations in reduction  $R$  (or,  $C$  under the result of Lemma 3).

*Hypothesis:* If the number of iterations  $n$  is 0, then the theorem follows trivially.

*Inductive Step:* We assume that if  $n = k$ , the theorem is true.

Consider the intermediate statement  $\tau$  translated by the Reduction algorithm in  $k + 1$  iterations. If  $\tau'$  is the intermediate statement after  $k$  iterations, then  $\tau$  differs from  $\tau'$  by having one additional rightmost quantifier and one less distinct OE term. Application of the Construction algorithm to  $\tau$  eliminates this rightmost quantifier and brings back the OE term (under the result of Lemma 1). Thus, one iteration of the construction algorithm applied to  $\tau$  gives us  $\tau'$ .

By induction,  $C^n(R^n(\zeta)) = \zeta$ . This completes the proof of the theorem.

*Completeness Theorem:* Given RFOPL statement  $\eta$ ,  $\eta$  can be translated into NL statement  $\zeta$ . Also  $\eta$  can be retranslated from  $\zeta$ , that is,  $R(C(\eta)) = \eta$ .

*Proof:* We show that  $R^n(C^n(\eta)) = \eta$  by induction on the number of iterations in reduction  $C$  (or,  $R$  under the result of Lemma 4).

*Hypothesis:* If the number of iterations  $n$  is 0, then the theorem follows trivially.

*Inductive Step:* We assume that if  $n = k$ , the theorem is true.

Consider the intermediate statement  $\tau$  translated by the Construction algorithm in  $k + 1$  iterations. If  $\tau'$  is the intermediate statement after  $k$  iterations, then  $\tau$  differs from  $\tau'$  by having one additional distinct OE term and one less rightmost quantifier. Application of the Reduction algorithm to  $\tau$  eliminates this distinct OE term and brings back the rightmost quantifier (under the result of Lemma 2). Thus, one iteration of the Reduction algorithm applied to  $\tau$  gives us  $\tau'$ .

By induction,  $R^n(C^n(\zeta)) = \zeta$ . This completes the proof of the theorem.

## Appendix C: Security Formulations for the State Transition Model

*Initial State:* The initial state is defined to be  $(\{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\})$  and trivially satisfies the state invariant SI stated in Section 4.4.

*Theorem 1:* If state  $S_n$  is the new state after application of a sequence of  $n$  state transition rules on state  $S_0$  and if  $S_0$  satisfies the invariant SI, then  $S_n$  also satisfies the invariant SI.

*Proof:* We show a proof of the theorem by induction.

Let the sequence of transition rules applied be  $\{R_1, R_2, \dots, R_n\}$  and let the resulting sequence of states be  $\{S_0, S_1, \dots, S_n\}$  such that state  $S_i$  results from an application of  $R_i$  on state  $S_{i-1}$ .

Consider the case where state  $S_{i-1}$  is known to satisfy SI. We have to show that the application of transition rule  $R_i$  results in a state  $S_i$  that also satisfies SI.

Hypothesis:  $I = 0$ .  $S = S_0$  – the initial state satisfies SI as shown above.

Induction Step:  $I > 0$ . We assume that state  $S_{i-1} = S = (CD, CC, CAM, CRes, JRes, CP, NCR, CCOM, HIS)$  satisfies the state invariant SI, that is, I1 ... I9 are satisfied by CD, CC, CAM, CRes, JRes, CP, NCR, CCOM and HIS. We show below that  $S_i$  satisfies SI for Rule R5 – Propose common shares resources. It can be similarly shown that other transition rules also result in  $S_i$  satisfying SI. (Proofs for Rules 10 and 11 are provided in [18]).

Application of rule  $R_5 =$  Propose common shared resources  $(CP, D_pProp = \{D_1O'', D_1App'', D_2O'', D_2App'', \dots, D_kO'', D_kApp'', JO'', JApp''\})$

$S' = (CD', CC', CAM', CRes', JRes', CP', NCR', CCOM', HIS') = (CD, CC, CAM, CRes, JRes, CP \cong D_pProp, NCR, CCOM, HIS)$

$(CD, CC, CAM, CRes, JRes, NCR, CCOM, HIS)$  satisfy I1, I2, I3, I4, I6, I7, I8 and I9.

CP satisfies I5 and  $D_pProp$  satisfies

- (i)  $\&Z = (D_jO'', D_jApp'') \subset D_pProp: Z \geq CRes$
- (ii)  $\&Z = (JO'', JApp'') \subset D_pProp: Z \geq JRes$
- (iii)  $D_pProp \models D_iC \cong GC$  where  $D_iC \subset CC, GC \subset CC$

Therefore,  $CP \cong D_pProp$  satisfies I5.

By hypothesis,  $S_0 = (CD, CC, CAM, CRes, JRes, CP, NCR, CCOM, HIS)$  satisfies SI. Thus, by induction  $S_1, S_2, \dots, S_n$  also satisfy SI. This completes the proof of the theorem.

*Theorem 2:* If  $S_0 = (CD_0, CC_0, CAM_0, CRes_0, JRes_0, CP_0, NCR_0, CCOM_0, HIS_0)$  and  $S_n = (CD_n, CC_n, CAM_n, CRes_n, JRes_n, CP_n, NCR_n, CCOM_n, HIS_n)$  are two states that satisfy SI and  $CD_n \cong HIS_n \neq CD_0, CC_n \cong HIS_n \neq CC_0, CAM_n \cong HIS_n \neq CAM_0, CRes_n \cong HIS_n \neq CRes_0, JRes_n \cong HIS_n \neq JRes_0, CP_n \cong HIS_n \neq CP_0, NCR_n \cong HIS_n \neq NCR_0,$  and  $CCOM_n \cong HIS_n \neq CCOM_0$ , then there exists a sequence of rules that transforms  $S_0$  to  $S_n$  and is secure.

*Proof:* We prove the theorem by contradiction

Let the sequence of rules  $\{R_1, R_2, \dots, R_n\}$  transform  $S_0$  to  $S_n$  through intermediate states  $S_1, S_2, \dots, S_{n-1}$  such that each  $S_{i+1}$  adds a set of elements to  $CD_i, CC_i, CAM_i, CRes_i, JRes_i, CP_i, NCR_i, CCOM_i,$  or  $HIS_i$ .

Let us assume that the above sequence of rules contains at least one insecure transform  $R_i$  ( $1 < i < n$ ) which, when applied to state  $S_{i-1}$  results in a new state  $S_i$ . There are 11 possible cases for transform  $R_i$ . We consider the following case for transform Rule 5, namely proposing common shared resources. The arguments for the remaining ten cases are similar.

Consider transform  $R_i$  to be Rule 5.  $S_{i-1} = (CD_{i-1}, CC_{i-1}, CAM_{i-1}, CRes_{i-1}, JRes_{i-1}, CP_{i-1} \cong D_pProp, NCR_{i-1}, CCOM_{i-1}, HIS_{i-1})$

Since  $R_i$  is not a secure transform, the *if* part of Rule 5 is not satisfied for the set of elements  $X = D_pProp$  in  $CP_i$ . Thus, there exists a set of elements  $X$  in  $CP_i$ , which violate I5 of SI. Therefore state  $S_i$  does not satisfy SI. Moreover,  $S_n = (CD_n, CC_n, CAM_n, CRes_n, JRes_n, CP_n, NCR_n, CCOM_n, HIS_n)$  may be reached from intermediate state  $S_i = (CD_i, CC_i, CAM_i, CRes_i, JRes_i, CP_i, NCR_i, CCOM_i, HIS_i)$ , thus  $CD_n \cong HIS_n \neq CD_i, CC_n \cong HIS_n \neq CC_i, CAM_n \cong HIS_n \neq CAM_i, CRes_n \cong HIS_n \neq CRes_i, JRes_n \cong HIS_n \neq JRes_i, CP_n \cong HIS_n \neq CP_i, NCR_n \cong HIS_n \neq NCR_i,$  and  $CCOM_n \cong HIS_n \neq CCOM_i$  (since in our model, all state transitions except for the *commit* and *leave coalition* transitions only allow augmentations of sets CD, CC, CAM, CRes, JRes, CP, NCR, CCOM, HIS and any reductions in the *commit* or *leave coalition* transitions are maintained in the HIS set). If  $CP_i$  contains the set of elements  $X$  which violates I5, then  $CP_n \cong HIS_n$  also contains the same set of elements  $X$  and hence we arrive at the conclusion that  $S_n$  does not satisfy SI, which contradicts our hypothesis.

Thus the sequence of rules which takes us from  $S_0$  to state  $S_n$  does not contain any insecure transform, and is, therefore, secure. This completes the proof of the theorem.

## Appendix D: Examples of Common Access State Negotiations

In this section we illustrate three examples of negotiating a common access state in the state-transition model. Negotiation constraints are specified in NL. The first example is one of negotiating airlines routes introduced in Section 2. In the second example we illustrate how our model supports coalition dynamics. Additional examples are provided in [18].

### D.1. Example 1: Negotiating Routes with Least-Privilege Constraints

This example was introduced in Section 2. We review the coalition objective, present the negotiation constraints in NL and describe the negotiation process using the state-transition model.

*Coalition Objective:* To share six route types (1 ... 6) between three domains  $D_1$ ,  $D_2$ , and  $D_3$  and to jointly own and administer an auditing application with access to shared routes.

*Negotiation Constraints:* The following two *global negotiation constraints* have been agreed upon:

- (1) Domains that have unique route types must share them (*obligation constraint*). As a consequence, Domain  $D_1$  must share route type 6 and Domain  $D_2$  must share route type 3.
- (2) Sharing of route types must minimize the number of routes shared (*least privilege constraint*); i.e., if two or more domains are capable of sharing the same route type, then the one that comprises the lowest number of objects will be used.

These constraints are specified in NL as follows:

Let  $DU$  be a list of all coalition users:  $DU = \{D_1U, D_2U, D_3U\}$

For each route type  $route_i$ , the following two lists are defined

$Dom_i = \{D_1, \dots, D_3\}$  – list of domains that have  $route_i$

$N_i = \{nroutes_1, \dots, nroutes_3\}$  – number of routes in each domain's route type

*Constraint (1)*  $length(Dom_i) = 1$  ♥

$route_i \subset coalition\ objects(coal.\ permissions(coal.\ roles(DU - D_jU)))$

where domain  $D_j \subset Dom_i$

*Constraint (2)*  $length(Dom_i) > 1 \langle list\_min(N_i, N_{ij})$  ♥

$route_i \subset coal.objects(coal.permissions(coal.roles(DU - D_jU)))$

where domain  $D_j \subset Dom_i$  and  $N_{ij}$  is the  $j^{th}$  element of list  $N_i$

#### The Negotiation Process

In Figure 4 below, we illustrate the negotiation process in the state-transition model for the above coalition objective and negotiation constraints. We show the relevant state variables in the common access state as the negotiation proceeds with application of the state transition rules.

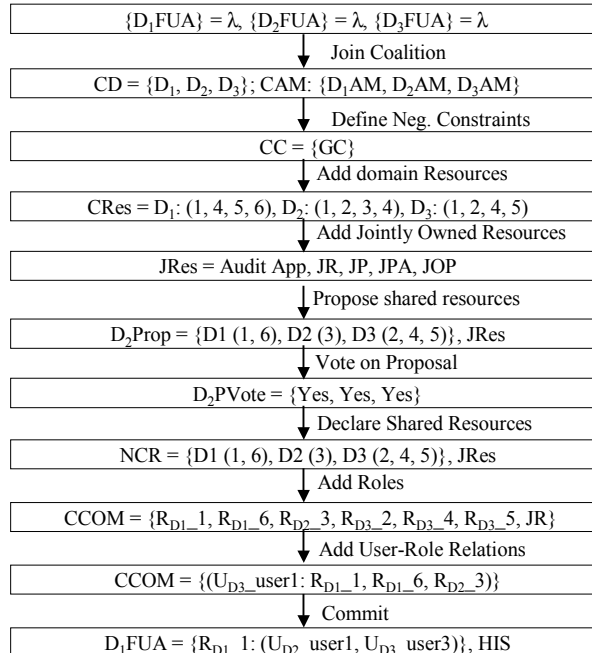


Figure 4: Example 1 - Negotiation Routes with Least Privilege Constraints

Here negotiation begins in a state where the domains do not share any resources. Domains join the coalition and agree on the above-mentioned global negotiation constraints. Domains then add their route types to the pool of coalition resources that may be shared and the audit application supported by RBAC access policies that is to be

jointly owned. Domain  $D_2$  proposes a set of shared resources that satisfies the global negotiation constraints. All domains unanimously vote for this proposal, which is then declared to be the negotiated set of coalition resources. If any domain had voted “no”, then negotiations would have proceeded with alternate proposals. Once the negotiated set of resources is declared, domains provide roles and users to access these resources. Here the domains provide one role per route type that they share with the coalition and they provide one user to be a member of all these (foreign) roles. The domains then commit this common access state by enrolling foreign users in the roles that have permissions for shared resources (including the jointly owned auditing application) after verifying that the state does not violate any of the local access constraints of the three domains. The figure shows the committed state for Domain  $D_1$  and those for the other two domains are similar.

#### D.2. Example 2: Supporting Coalition Dynamics

In Section 3 we have discussed three possible coalition dynamic events, namely, domain join, voluntary domain departure, and involuntary domain departure. A domain join event is supported in the model by the joining of a new domain at the beginning of the process of re-negotiating a common access state. The set of coalition domains, including the new one, re-negotiates the common access state where member domains can retain the previous sharing agreements simply by including them in the proposal from the state history variables. The proposal will also include the joining domain’s resources. The domains then vote on this proposal and commit it after adding roles and users in a manner similar to the first negotiation. A voluntary domain departure is also easily supported by the state transition rule *leave coalition* where the departing domain leaves the coalition by withdrawing its privately owned resources, that is, by revoking permissions of foreign users to these resources. The remaining domains would then choose re-negotiate the common access state if (1) there are jointly owned resources in which case they would re-negotiate access policies for these resources, or (2) the departing domain withdraws any resources essential to the coalition objective in which case the domain would redefine the coalition objective and negotiate a common access state that satisfies this objective.

We now illustrate how the model supports the involuntary departure of a domain with following agreement established prior to coalition setup: if  $n-1$  domains decide to exclude the  $n^{\text{th}}$  domain from the coalition, then they can do so as long as they allow the departing domain to withdraw any resources it had ever contributed for joint ownership (in addition to its privately owned resources). In Example 2, if after coalition setup, Domains 1 and 3 decide to exclude Domain 2 from the coalition then would re-negotiate the common access state as follows.

*Coalition Objective:* Exclude Domain 2 from the coalition while adhering to agreement on involuntary domain departure.

*Negotiation Constraints:* The domains agree to the following global negotiation constraint, which captures the agreement on involuntary domain departure.

- If a proposal is to be accepted with only  $n-1$  votes for a known set of  $n$  domains, then the common access state cannot include any jointly owned resources that were ever contributed by the excluded domain.

This constraint would be defined in NL as follows.

Let  $\text{Joint\_dom2}$  be the list of jointly owned resources that were contributed by Domain 2 in all prior negotiations.

Here  $\text{Joint\_dom2} = \{\text{Application 2}\}$

- Constraint:  $|\text{JApp} \sim \text{Joint\_dom2}| = \pi$ , where  $\text{JApp} \subset \text{CAM}$

#### The Negotiation Process

In Figure 5 below, we illustrate the negotiation process in the model for the above coalition objective and negotiation constraints. We show the relevant state variables in the common access state as the negotiation proceeds with application of the state transition rules. For this negotiation, the *majority* variable for security invariant I7 and transition Rule 7 (Declare negotiated set of resources) is set to  $n-1$ , where  $n$  is the number of domains in the coalition when negotiation begins.

Here negotiation begins in a state where the domains share resources as negotiated in the common access state in Example 2. Domains 1 and 3 then join the coalition and agree on the above-mentioned global negotiation constraint. Then domains add the applications 1 and 3 to the pool of applications that are to be jointly owned and administered. Domain  $D_1$  proposes the joint ownership of these resources, excluding application 2 to satisfy the negotiation constraints, and this proposal is accepted by Domain  $D_3$ , that is, by a *majority* of  $n-1$  domains. Then the domains declare this as the negotiated set of resources. Domains  $D_1$  and  $D_3$  then provide one user each for membership to the roles for jointly owned application 1 and 3. The common access state is then committed by enrolling these users to the roles if it satisfies the local access constraints of Domains  $D_1$  and  $D_3$ .



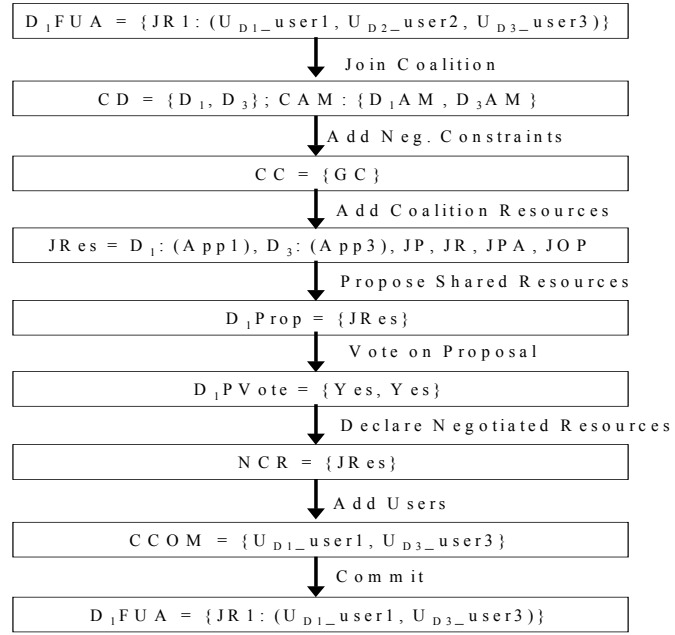


Figure 5: Example 4- Supporting Coalition Dynamics