Sharif University of Technology
Department of Computer Engineering

NORMATIVE LOGIC BASED SEMANTIC-AWARE AUTHORIZATION MODEL

Morteza Amini

A Thesis Submitted in Partial Fulfilment of the Requirements
for the Degree of Doctor of Philosophy

January 2010

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Rasool Jalili) Principal Adviser

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Ali Movaghar)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Hassan Mirian)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Mohammad Ardeshir)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Saeed Jalili)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Mehran Soleiman Fallah)

Approved for the University Committee on Graduate Studies

To my kind parents who raised me and supported me by their prays, and I owe them all of my successes.

To my dear wife who always encouraged and emotionally supported me, and tolerated all the problems of doing this thesis.

To my lovely daughter, Kiana, with her beautiful smile.

Acknowledgements

First and foremost, I must pray God for giving me the power and ability to do this research.

I would like to cordially thank Dr Jalili, my advisor, who did everything for doing this thesis in the best way. I must thank him for his guidance, patience, and encouragement over these years.

I would also like to express my gratitude to Prof. Mohammad Ardeshir (from Department of Mathematical Science of Sharif Univ. of Tech.), and Dr Mehran Soleiman Fallah (from Department of Computer and IT of Amirkabir Univ. of Tech.) for their useful comments and guides during this research project, and reviewing this thesis. My appreciate also goes to the internal reviewers of my thesis, Prof. Ali Movaghar and Dr Hassan Mirian for their comments and feedbacks on my annual progress reports. I am also thankful to Dr Saeed Jalili to accept to review my thesis.

I am grateful to my friends and colleagues Amir Reza Masoumzadeh, Sara Javanmardi, Hassan Takabi, Ali Noorollahi, Jafar Haadi Jafarian, Mohsen Taherian, and Zeinab Iranmanesh for all of the cooperations that we had together to promote access control research field in our laboratory. I especially like to thank Fathieh Faghih and Mousa Amir Ehsan for their significant efforts in developing the security agent prototype for this thesis.

Abstract

Semantic technology provides an abstraction layer above existing computational environments, especially the Web, to give information a well-defined meaning. Moving toward semantic-aware environments imposes new security requirements. One of the most important requirement is the authorization and security policy inference based on the existing semantic relationships in the abstract (conceptual) layer. Most of the authorization models proposed for these environments so far are incomplete and their inference rules are not guaranteed to be consistent, sound, and complete. To have a sound and complete system for policy specification and inference, in this thesis, a family of modal logics, called MA(DL)^2, is proposed with the corresponding syntax, proof theory, and semantics. The core of this family of logic is a combination of multi-authority version of deontic logic (MADL) and description logic (DL). It is proven that the proposed logics in this family are sound, complete, and decidable and have finite model property. We then propose an authorization model based on the MA(DL)^2 logic, which enables authorities of different security domains to specify their security policies in conceptual and ground (individual) levels in terms of deontic statuses: permission, obligation, and prohibition. The logical foundation of the model enables it to infer implicit security policies from the explicit ones based on the semantic relationships defined in subjects, objects, and actions ontologies. Cooperative security management in shared subdomains (in spite of the distribution of policy specification), context-awareness and modal conflict resolution of policies are the other characteristics of the proposed model. To show the applicability of the proposed model, an automatic inference method based on the analytical tableaux approach is presented for the MA(DL)^2 logic and has been implemented in Prolog as well. Using the developed inference engine, a prototype of an authorization and access control system has been implemented and experimental results of its evaluation is presented in the thesis.

Keywords: Data Security, Authorization Model, Access Control, Deontic Logic, Description Logic, Semantic Technology, Semantic-Aware Environment

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The shift from current computing environments to the semantic-aware ones (e.g., Semantic Web, Semantic Grid, and Semantic Cloud Computing Environments) takes aim at giving information a well-defined meaning (semantics). The main goal of giving semantics to information is making machines capable of interpreting and processing them. This has been yielded by enriching the existing information and resources with an abstract layer (named *ontology*), which specifies the shared conceptualization.

Moving to the semantic-aware environments (call in short SAEs), which are the new generation of distributed environments where semantic technology is used, imposes new security requirements [31, 78], which should be considered in proposing an appropriate authorization model for them. The most important drawback of the traditional authorization models and the new proposed models for SAEs is that they do not cover all the security requirements of such environments, especially the capability of policy specification in the abstract (conceptual) level and the ability of policy inference based on the semantic relationships specified between the entities. In fact, most of the existing models use policy inference rules, which are not guaranteed to be consistent, sound, and complete. A proper solution to satisfy this requirement is using a logic with proved properties for authorization in SAEs.

In this thesis, we are about to introduce a new authorization model based on a new logic to provide a sound and complete system for specification and inference of security policies in the both conceptual and ground (individual) levels, and to cover all the essential security requirements of SAEs. In this model (called MA(DL)^2-AM), a logical language (named MA(DL)^2), which is proposed in this thesis, is employed for specifying and inferring the security policies.

MA(DL)^2 is a family of normative logics, which its core is the combination of multi-authority (poly-modal) version of standard deontic logic and description logic. Using this logic, administrators (authorities) of different security domains can specify their security policies in terms of normative statuses; obligations, prohibitions, permissions, and gratuities. Using this logic enables us to take the impact of semantic relationships between the classes of entities on inference of security policies into account, and thus, infer the implicit security policies from the explicit ones. Formal semantics of this logic family and the proofs of soundness, completeness, and decidability of the proposed proof theory guarantee the correction of policy inference in the proposed authorization model. Furthermore, an analytic tableaux method for automated reasoning in MA(DL)^2 is introduced in this thesis, which has been used for practical implementation of an inference engine. To show the applicability of the proposed model, a prototype of an access control system based on MA(DL)^2-AM by using MA(DL)^2 inference engine has been developed.

1.1 Contributions of the Thesis

In brief, the main contributions of this thesis are as follows:

1. Proposing an overall framework for authorization and access control in distributed SAEs, and cooperative security management for shared security domains.

2. Proposing MA(DL)^2 normative logic family (as a combination of deontic logic and description logic) for security policy specification and inference in the conceptual level and introducing security knowledge base.

3. Proving the soundness, completeness, and decidability, as well as investigating the expressive power and computational complexity of the MA(DL)^2 logic family.

4. Presenting an analytic tableaux method for the MA(DL)^2 logic family and implementing an inference engine based on the presented method in Prolog.

5. Proposing a formal authorization model for SAEs (called MA(DL)^2-AM) based on the MA(DL)^2 logic family. Using the model we can specify security policy rules in both conceptual and ground (individual) levels, and infer implicit policy rules based

on the semantic relationships defined on the elements participating in authorization and access control.

6. Implementing a prototype of an access control system (as a security agent) based on the MA(DL)^2-AM authorization model.

1.2 Thesis Structure

In the rest of this thesis, Chapter 2 presents preliminaries of information security, and surveys and classifies the classical authorization models as well as the authorization models proposed based on different logics.

In Chapter 3, semantic technology and semantic-aware environments (SAEs) are introduced. Furthermore, security and authorization requirements of SAEs are listed in this chapter. By surveying the related work for modeling authorization for these environments, we present the model of the computational system in SAEs, and also our overall framework for authorization and access control in these environments. In this overall framework, we show how to use our proposed logic for authorization and access control.

Chapter 4, after providing a brief introduction to description logic and standard deontic logic, presents the members of our proposed logic family, i.e., MA(DL)^2, by their syntax, Kripke semantics, and Hilbert style proof theory.

The properties of the MA(DL)^2 logic family are investigated in Chapter 5. The proofs of soundness, completeness, and decidability of the members of the MA(DL)^2 logic family, and discussion about the computational complexity of satisfiability problem and expressive power of the proposed logics are presented in this chapter.

In Chapter 6, we introduce our proposed authorization model for SAEs. The model is called MA(DL)^2-AM, and is based on the MA(DL)^2 logic family. The formal specification of the model and its elements as well as the precise description of conflict resolution and access control procedure are presented in this chapter.

The implementation of a prototype of an access control system (serves as a security agent in the proposed overall framework) and the experimental results are reported in Chapter 7. For this purpose, we introduce an analytic tableaux system for automating the inference in the proposed logic and implement the required inference engine based on the introduced method for access control.

Chapter 8 concludes the thesis and draws some future work related to this thesis.

Chapter 2

Authorization and Logic

Specification of security policies is the essential requirement to keep the resources in a computing environment protected and secure. For this purpose, we often follow an authorization model appropriate for our application. In the rest of this chapter, after providing some definitions, the authorization models proposed for different environments are classified and surveyed. Using logics for authorization in distributed environments has many advantages, which are investigated in this chapter. The survey of some related work on using logics for policy specification and authorization are also presented in the rest.

2.1 Preliminaries

When we talk about authorization models and mechanisms, we need to first define precisely the meaning of the concepts existing in this domain. Therefore, we first define security policy, model, and mechanism. Then, authorization and access control are defined. The Definitions of trust and trust management ends this section.

2.1.1 Security Policy, Model, and Mechanism

Security Policy: a security policy defines the security requirements of an organization, and the steps to reach a proper level of security. A security policy in a system, partitions its states into secure (authorized) and nonsecure (unauthorized) states [28, 35].
Security Model: a security model is an abstraction of a security policy. A security model formally specifies the relations between the subjects, objects, and other elements of a system.

In other words, a security model is a formal specification of a security policy [28].

Note that access control model, authorization model, and protection model have the same meaning as security model in the literature (e.g., see [20, 100, 77]). In this thesis, based on the definition we presented for authorization, we prefer to use *authorization model* for our purpose, i.e., a model for security policies restricting or forcing accesses to the resources.

Security Mechanism: a security mechanism is a method, tool, or procedure for enforcing a security policy [28]. Security mechanisms are divided into the three categories; prevention mechanisms (access control systems and firewalls), detection mechanisms (e.g., intrusion detection systems and honeypots), and recovery and retrieve mechanisms (e.g., audit and backup systems).

Identification & Authentication (I&A): identification is the way that a user identifies herself to the system (e.g., by her user name), and authentication is the process in which the system authenticates that the identified user is the subject claims to be [28]. Authentication is the prerequisite of may other security mechanism such as access contol [149, 138].

2.1.2 Authorization and Access Control

There are different definition for authorization and access control. Some of them define the two equal [4, 138]; however, some of them define one of them including the other one [45, 99, 128]. The definition seems more acceptable for these two words is as follows. In fact, we distinguish the policy specification stage from the policy enforcement stage.

Authorization: authorization is the specification of security policy and determining the access rights to the resources [89, 97, 93].

Access Control: access control is the enforcement of security policies, and the process that controls which subjects, under which conditions can have which accesses to the resources [149].

2.1.3 Trust and Trust Management

There are different meanings for *trust* in computer science that makes difficult to present a common acceptable definition [115]. The two common definitions for trust are named *reliability trust* and *decision trust* [95].

Reliability trust is defined as “the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends". However, decision trust is defined as "the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible".

Based on the definition of trust, there are different types of trust including provision trust, access trust, delegation trust, identity trust, and context trust. In this thesis, by *trust* we mean *access trust*, and by *trust management* we mean *distributed authorization* [103].

2.2 Authorization Models

Since 1971 such that the first authorization model (i.e., the access matrix model) was proposed by Lampson [100], different authorization models have been proposed. The variety of the models is resulted from the variety of security requirements identified for different environments. Thus, we can classify the proposed authorization models based on different factors as depicted in Figure 2.1. In this figure, the second level shows the classification factor and the third level shows the resulted classes by the classification factors. In the rest of this chapter, we survey some of the important classes.

2.2.1 Classification based on the Right Assignment Approach

In most of the references, classification of authorization models are made base on the way that the access rights are assigned to the users or subjects [47, 143]. In the preliminary classification presented in Orange Book [37], authorization and access control models are classified into discretionary access control (DAC) models, and mandatory access control (MAC) models. After introducing role-based access control (RBAC) models, it has been extended to three classes of models; i.e., DAC, MAC, and RBAC.

In DAC models, an individual user (e.g., the owner) can set accesses over the resources, and authorization is on the discretion of the users (owners) [28]. In such models, access rights are defined based on the identity of the subject and the identity of the objects involved. Flexibility of DAC models, make them suitable for the spread range of applications, especially the ones used in commercial and industrial environments. The main drawback of these models is that they cannot control information flow where multi-level security is required [35, 143].

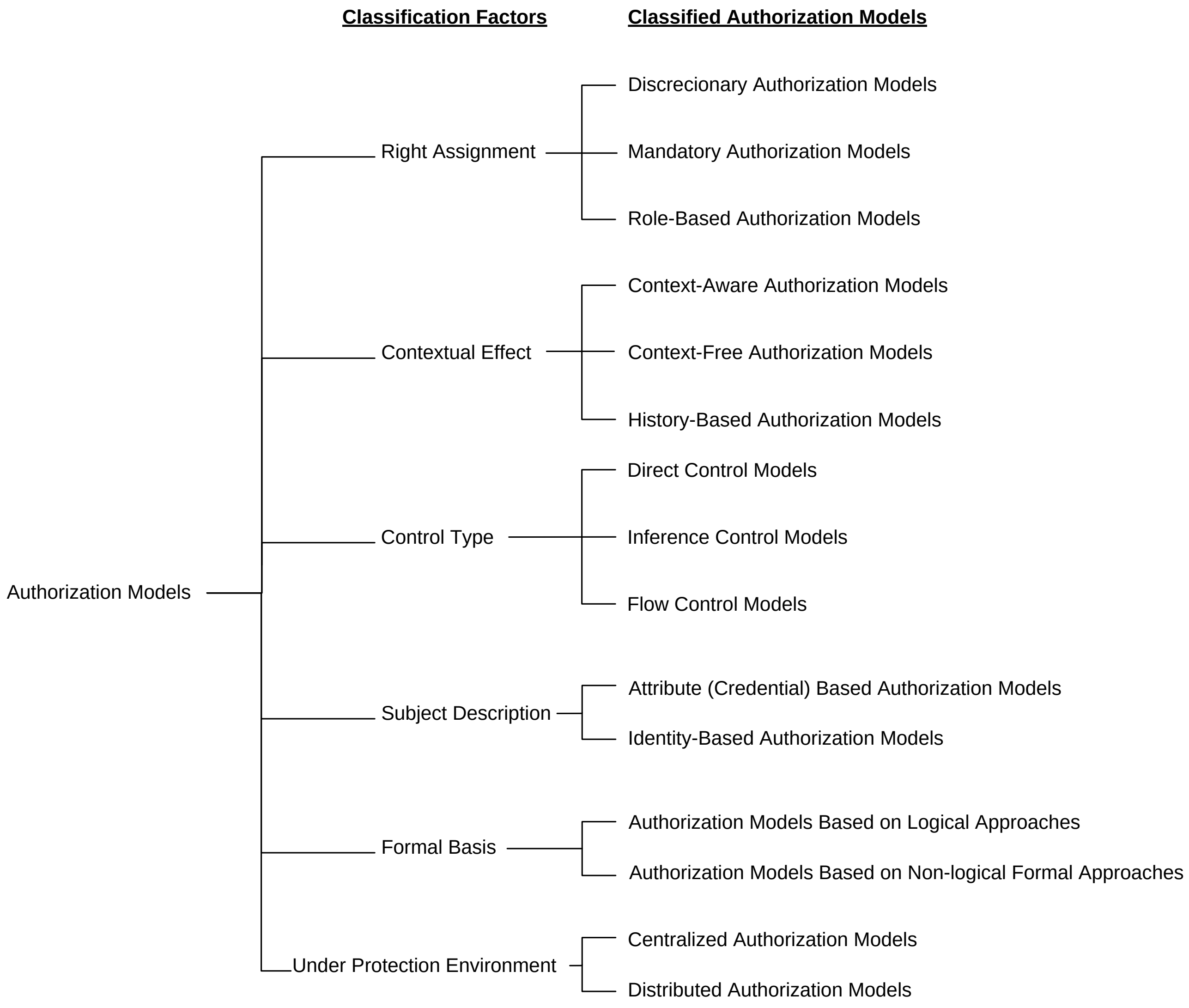

Classification Factors
Classified Authorization Models
Authorization Models
Right Assignment
Discrecionary Authorization Models
Mandatory Authorization Models
Role-Based Authorization Models
Contextual Effect
Context-Aware Authorization Models
Context-Free Authorization Models
History-Based Authorization Models
Control Type
Direct Control Models
Inference Control Models
Flow Control Models
Subject Description
Attribute (Credential) Based Authorization Models
Identity-Based Authorization Models
Formal Basis
Authorization Models Based on Logical Approaches
Authorization Models Based on Non-logical Formal Approaches
Under Protection Environment
Centralized Authorization Models
Distributed Authorization Models

Figure 2.1: Classification of authorization models based on the different factors.

There are many DAC models. Some famous models in this class are the Lampson's matrix model [100], the HRU general model [77], the schematic protection model (SPM) [139] and ESPM [8] as its extended version, the typed access matrix model (TAM) [140] and dynamic TAM (DTAM) [148], the take-grant model (TG) [107], and the ACTEN model [61].

In MAC models, accesses are restricted based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity [37]. In such models, users cannot alter the accesses, and access requests are granted based on the axioms defined in the model. One of the important goals of these models is information flow control to guarantee confidentiality and integrity of classified information. Therefore, MAC models are more suitable for military environments.

The famous MAC models proposed for preserving confidentiality and/or integrity are BLP [20], the Biba integrity model [27], the Dion integrity and confidentiality model [53], and the Denning's lattice-based model [51]. There are also some MAC models for multi-level databases such as the Sea-View model [52], the Jajodia-Sandhu model [86], and the Smith-Winslett model [147]. Samples of the MAC models proposed for commercial and financial area are Clark-Wilson [39] and Chinese Wall policy [33], which concentrate on conflict of interests and separation of duties. Considering context in mandatory policy specification is a new approach, which is considered recently in [85, 84].

In the RBAC models, which are proposed by Ferraiolo and Kuhn [57] for easier administration of permissions in organizations, the permissions are associated to roles, and roles are assigned to users. Each role is assigned the permissions required to do its defined functions and transactions. Sandhu [142] introduced the RBAC family of reference models, in which RBAC_0 is the base model that satisfies the above description. We can have role-hierarchy similar to the one exists in the organization, and roles can inherit permission from other roles. RBAC_1 is RBAC_0 plus role-hierarchy. RBAC_2 is obtained by adding constraints, such as separation of duty (SoD), to the base model. RBAC_3 is the combination of RBAC_1 and RBAC_2. These models are specified as a standard by NIST [58] and ANSI [10]. Role-Graph [126] is another role-based authorization model, in which three graphs for user groups, roles, and permissions can be defined. Role-based administration of roles is also proposed for RBAC models as the ARBAC97 model [141].

RBAC models proposed by Sandhu [142], have been more considered in recent years in comparison with the other classical models. RBAC has been extended by defining subjective, objective, and environmental roles in GRBAC [41]. Adding temporal constraints to RBAC results in TRBAC [22]. Considering context and contextual constraints resulted in dynamic context-aware models, DRBAC [164] and CARBAC [145].

2.2.2 Classification based on the Subject Description Approach

Most of the old and traditional authorization models are *identity-based*, such as HRU [77], BLP [20], and RBAC [142]. In such models there is an assumption that users (subjects) are known and can be predefined. Identity-based authorization models are not suitable for open and distributed computing environments, as huge number of users exist and most of them are unknown for the system [157].

Contrary to the identity-based authorization models, we have *attribute-based* or *credential-based* authorization models, where subjects are described based on their attributes not their identities. In such models, users provide some credentials or attribute certificates to prove that they are eligible to access the resources. Thus, policy rules are defined based on the users' attributes not their identities. For this purpose, ITU-T in X.509 version 4 [83], besides describing public-key infrastructure (PKI), introduces privilege management infrastructure (PMI), which provides an infrastructure for issuing and validating the attribute certificates.

2.2.3 Classification Based on the Formal Basis of Authorization Models

Precise and unambiguous specification of security policies is so important to have a trustworthy secure system. Using formal methods (e.g., logics) in security modeling provides a satisfactory answer to the above requirement. In formal specification of policies in authorization and security models, the following two approaches are mostly used:

- Using non-logical approaches and languages such as set theory. Samples of the researches pursuing this trend can be found in [20, 99, 10, 113].

- Using logics like first-order logic, stratified logic, and different kinds of modal logics. Samples of the valuable researches following this trend are [4, 159, 88, 104].

As we concentrate on using logic for policy specification in our proposed authorization model, in the rest of this chapter, more details on the benefits of using logics for authorization are presented. The different kinds of logics, which are used for this purpose as well as the authorization and security models proposed based on these logics are also surveyed.

2.3 Logics for Authorization

Authorization and access control in centralized environments are not a very difficult problem; however, in distributed environments it becomes a complicated problem due to the characteristics of such environments. Characteristics such as broadness, distribution of resources, autonomy of the agents and components, and heterogeneity. Such complexity and difficulties impose new problems, which can be mostly answered using logics, which provides abstraction, preciseness, expressive power, and inference ability in policy specification and enforcement.

2.3.1 Logics for Authorization in Distributed Systems

Authorization in closed centralized systems is different from authorization in open distributed systems. The differences can be found in the following aspects [32, 104]:

- *What does need protection?* In centralized environments, a central server, which is trusted by all users and agents, contains the required resources, and controls the accesses to them. However, in distributed environments, there are different servers and agents having valuable resources and we can not trust all of them easily.
- *Whom to protect against?* In open and large distributed environments, all the users are not known in advance and their access requests are not predictable. Therefore, the solutions based on the identity of users or the ones require the previous knowledge from the users are not applicable in these environments. Here, we need to use the attributes of the subjects and the related credentials in policy specification.
- *How to specify what is authorized or forbidden?* Due to the variety of the situations and requirements in heterogenous systems in distributed environments and the requirement of precise and unambiguous specification of security policies, we need to have an

abstract language, independent from implementation with proper expressive power to cover all the circumstances. Also the specifications must be enforceable efficiently, and there is trade-off between expressive power and implementability. Note that most of these requirements do not suit the centralized environments.

- *How to prove that the systems or applications are secure w.r.t. our definition for security?* Although in both centralized and distributed environments, the verification of security properties is required; in distributed environments, this is more complicated, and requires to employ formal and automated methods for policy specification and verification of the required properties.

Following the above discussion on authorization in distributed environments, it is clear that logics with the formal foundation, and acceptable expressive power and abstraction level, is a proper candidate to be used for authorization in such environments. The inference ability of logics can make a great help in decision making based on the facts collected from different resources in a distributed environment.

In the following sections, after discussing the advantages and disadvantages of using logics for authorization, we survey the related work on this issue.

2.3.2 Pros and Cons of Using Logics for Authorization

The advantages of using logics for security and especially authorization and access control are as follows [4, 103, 3, 32]:

1. Having a precise, clear, and unambiguous foundation, hence formal guarantees;
2. Flexibility for using in different applications and cases;
3. Expressiveness, which enables specifying different types of policies;
4. The abstraction of logic and independency from implementation, which enables composition of security policies for heterogeneous distributed environments;
5. Inference ability, which enables inferring implicit policy rules from the explicit ones based on the logical relationships existing in the environment;

6. Declarativeness, so that users do not need to have programming skills to use logics in different applications;
7. Providing a basis for verification of the required properties (e.g., consistency) in specified systems and policies; and
8. Provides many predesigned algorithms and tools (with the known computational properties) that can be used for security and authorization purposes.

The drawbacks of using logics is its difficulty and illegibility for the users with lack of logics literacy. This problem can be solved by having syntax similar to natural language, or representing formulae graphically. The other mentioned disadvantage is the impossibility of providing an efficient implementation for some of the proposed logical approaches in security and authorization [3, 47].

2.3.3 Related Work in Using Logics for Authorization

Using variety of logics to tackle the problems of specifying and proving the security of distributed systems started since 1988 by Glasgow [65]. Various sorts of logic including first-order logic, stratified logic (such as Horn clauses), modal logic (including temporal logic, epistemic logic, and deontic logic), and non-monotonic logic (such as default logic) have been used to model user beliefs and inference abilities, specification of security policies, context and temporal constraints, and historical interactions. They have also been used to verify the security properties of a system based on automated reasoning techniques. Since SAEs are distributed environments that are constructed over a logical foundation (i.e., description logic), and also fitness of logical authorization models for them (due to many reasons mentioned in previous sections), we briefly survey the logical security and access control models proposed for distributed environments.

The access control logic proposed by Abadi, Lampson, and others [4], provides a logical system for specifying composite principals, access control lists, and access delegation in distributed systems. The first attempt in providing a general framework for authorization made by Woo and Lam [159]. They proposed using default logic, which is a kind of non-monotonic logic, to specify authorization policies. Undecidability of the logic proposed by Woo and Lam, encouraged Jajodia, *et al.*, [88, 87] to define an authorization specification language (ASL) based on the stratified first-order logic, which is not only decidable but also linear. The proposed language supports the concepts of groups and roles and allows different rules, such as integrity, derivation, and conflict resolution that are regulating the access control decisions. The flexible authorization framework that is provided on ASL enables security officers to specify different types of policies such as Chinese wall policy, static and dynamic separation of duty, and positivie/negative take precedence policy.

Barker, *et al.*, [19] took a similar approach in specification of multiple types of policies (with emphasis on RBAC policies) using stratified the Horn-clauses logic. Freedom in using constrained negation in this language in comparison with ASL [87], encouraged the authors to leverage a partial-deduction approach for specializing access control on deductive databases [18]. The results illustrates that specialization procedure significantly improves the access request evaluation time.

Kushik, *et al.*, [98] introduced a constraint logic programming (CLP) based framework to determine access to partial or full ontologies in order to preserve confidentiality. In this framework, policies are CLP program (that are stratified Horn clauses with constructive negation) that prevent disclosing sensitive portions of an ontology. In the proposed approach for access control, the names of concepts or relationships are hidden while disclosing the overall structure of the ontology, and they are replaced with desensitized names.

Security policies may be changed over time. Thus dynamic policies are needed to specify authorized time-dependent behavior of a system. To this end, different policy languages proposed based on temporal (tense) logic. Bertino, *et al.*, introduced Temporal Authorization Base (TAB) [23] in which users and objects are fixed, and temporal authorizations are obtained by labeling authorization rules by periodic expressions and temporal operators i.e., `WHENEVER`, `ASLONGAS`, and `UPON`. Based on this idea, Joshi and others [94] proposed Generalized Temporal RBAC model (GTRBAC) that allows defining periodic as well as duration constraints on roles, user-role assignments, and role-permission assignments besides temporal role enabling constraints.

Considerable efforts made by Cuppens and others in specifying mandatory confidentiality policies for information flow control from 1990 to 1996 [42, 43, 44]. They proposed the usage of deontic logic in combination with other types of modal logic like Epistemic logic (to represent user's knowledge) and Temporal logic (to represent dynamic progress of system over time) to specify what a user is permitted to know. Although, deontic logic is capable of specifying and inference of obligations as well as access policies in discretionary models, Cuppons *et al.*, used it just in mandatory models (where multi-level security is satisfied). In this thesis, we leverage deontic logic accompanying with description logic to provide a discretionary authorization model for SAEs.

In spite of the fact that the surveyed logical models satisfy a wide range of security requirements of distributed environments (e.g., soundness and consistency proof, inference capability, contextual constraint specification, and inter-operability in distributed environments), none of them covers the essential requirements of SAEs, i.e., the effect of semantic relationships on policy propagation.

2.4 Summary

In this chapter, a taxonomy of authorization models is presented. Classification of authorization models are made based on the different factors. We briefly surveyed classical authorization models and the models proposed for distributed environments using different kinds of logics. In the surveyed logical approaches, using deontic logic for authorization seems more interesting. In deontic logic, permissions, prohibitions, and obligations are considered together, and their logical relationships (which are so important for conflict resolution) are taken into account. Also the capability of this logic in specifying the ideal situations (here means secure states which are specified by security policies) besides the real and actual ones (the situation that currently system has, which might be insecure), makes it more suitable for policy specification and enforcement. Following the above discussion and considering the fact that this logic can be easily combined with other logics, deontic logic has been chosen as a basis for the logic we have proposed for policy specification in inference for semantic-aware environments.

Chapter 3

Authorization in Semantic-Aware Environments

Bringing semantic technology to the current computational environments gave a well-defined meaning to the underlying information and resources. Presenting, sharing, and utilizing such a defined meaning and knowledge (in an abstract conceptual layer), make machines capable of interpreting, processing, discovering, and reasoning over the information and services.

The shift from the current computational environments to the semantic aware environments such as Semantic Web imposes new security requirements, which should be considered in a proper authorization model for them.

In this chapter, after introducing semantic technology and semantic-aware environments, the authorization requirements for these environments are presented. By a quick survey on the related work, and evaluating them based on the listed requirements, an overall framework for authorization in these environments are proposed at the end of this chapter.

3.1 Semantic Technology and Semantic-Aware Environments

Preceding of appearing semantic technology, almost all the information had to be interpreted by human to be of any use. Semantic technology enables machines to interpret shared information with the same meaning.

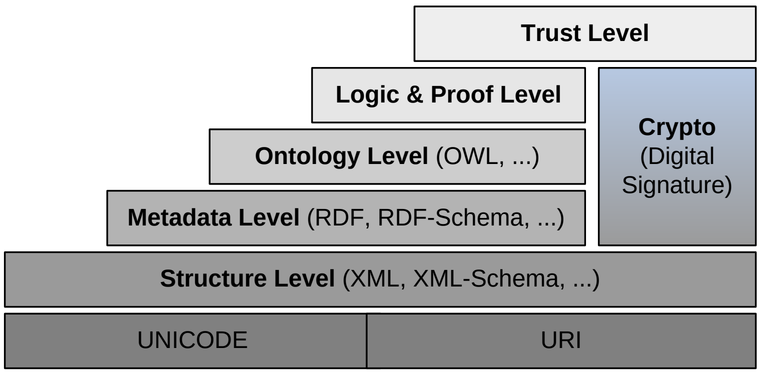

Trust Level
Logic & Proof Level
Crypto (Digital Signature)
Ontology Level (OWL, ...)
Metadata Level (RDF, RDF-Schema, ...)
Structure Level (XML, XML-Schema, ...)
UNICODE
URI

Figure 3.1: Semantic layer cake [66].

Definition 3.1.1 (Semantic Technology) *By Davis's definition [48],* semantic technology *is "a collection of software standards and methodologies that are aimed at providing more explicit meaning for the information in a computational environment".*

Semantics in semantic technology is interpreted as the *mapping between the structured data or concepts and the set of objects.* In this way, it provides a common understanding and interpretation about the concepts for machines and humans [48, 38].

Semantic technology encodes meanings (semantics) in an abstract layer separately from data and content files, and separately from application code, i.e., by describing the structure of the knowledge we have about them. With semantic technology, adding, changing and implementing new relationships between the entities or interconnecting programs in a different way can be just as simple as changing the external meta-data that these programs share.

The *semantic layer cake* (see Fig. 3.1) depicts a framework to present a variety of approaches to provide the meta-data that describe the semantics or meaning of underlying resources and information understandable by machines and humans [66]. The most underlying layer identifies any resource (information) that can be identified with a Uniform Resource Identifier (URI), e.g., animals, people, places, ideas, and actions. Upper layers provide vocabulary for describing classes, their properties, and their relationships with each other. More upper layers provide more vocabulary and thus more meaning (semantics) [66, 162].

A run-time semantic data model, which is more often leveraged to represent and share knowledge in a distributed world using semantic technology, is called *ontology* and defined as follows.

Definition 3.1.2 (Ontology) *Gruber [73] defined an ontology as "a data model that represents a domain and is used to reason about the objects in that domain and the relations between them".*

In short, an ontology is a specification of a conceptualization, and it is generally defined using constructors for:

- individuals (the basic or ground level objects or instances which may include concrete objects as well as abstract objects);
- concepts (classes, sets, collections, or types of objects that may contain individuals, other classes, or a combination of both);
- attributes (properties, features, characteristics, or roles that objects can have and share); and
- relations (relationships between objects or their corresponding concepts).

A family of formal languages that are used in recent years to encode and reason about an ontology (as a knowledge base) is the *description logic (DL)* family [13]. Figure 3.2 shows different knowledge representation languages and semantic models, and compares them with each other, from the reasoning capability and expressive power aspects. The most important characteristic of the languages belong to the DL family in comparison with their precedence (such as semantic networks) is their equipment with the logic-based semantics that provides a clear foundation and common understandable knowledge about them. Another distinguished feature is the ability of reasoning that allows to infer implicit knowledge about concepts and properties from the explicitly represented knowledge automatically.

OWL [49] is the standard language for representing ontology that is recommended by W3C. It provides three increasingly expressive sublanguages including OWL-Lite, OWL-DL, and OWL-Full, where OWL-DL is the most applicable one. In fact, OWL-DL is developed based on the $\mathcal{SHOIN}(\mathbf{D}_n)$ description logic.

By the above introduction to semantic technology and ontology, we define a *semantic-aware environment (SAE).*

Definition 3.1.3 (Semantic-Aware Environment (SAE)) *A semantic-ware environment (SAE) is a distributed environment where sematic technology is leveraged to provide meaning (semantics) for the entities, resources, and information existing in them.*

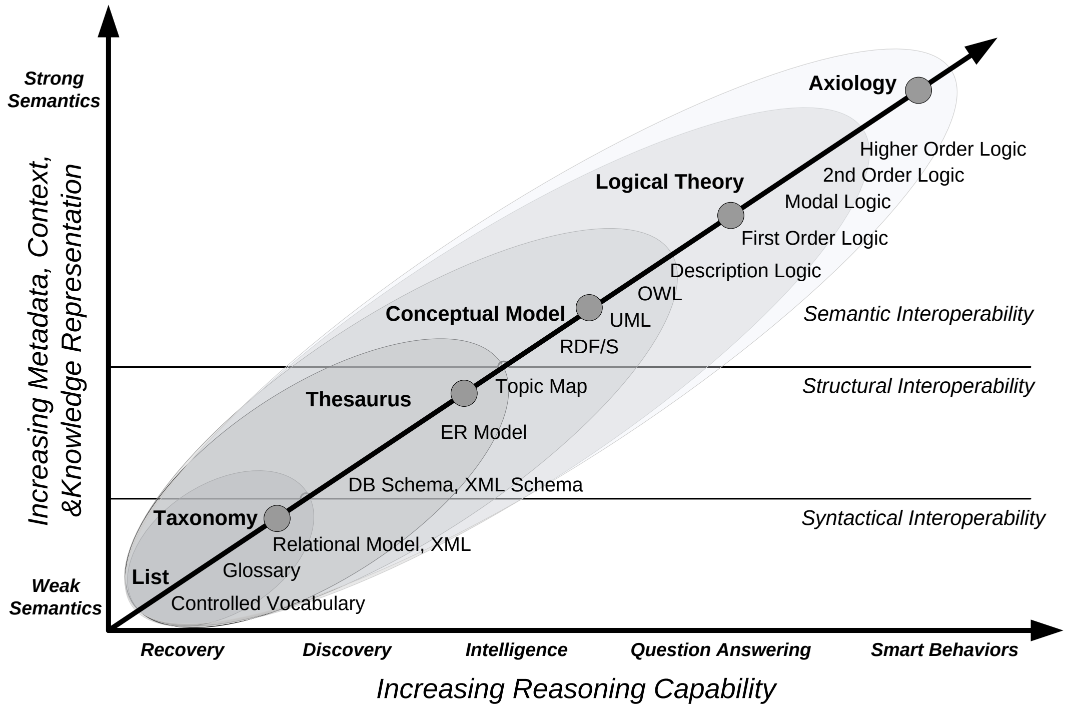


Figure 3.2: Comparison of semantic languages and data models [48].

The well-known examples of SAEs are Semantic Web, Web 3.0, Semantic Grid, and Semantic Cloud Computing Environments [48]. The common characteristics of SAEs can be listed as follows.

- There exists a common conceptualization specified by a semantic data model such as ontology.
- Agents and machines are intelligent in processing information. Intelligence means they can aggregate, integrate, and interpret diverse data sources; and do reasoning based on the defined semantic relationships.
- Such environments are distributed. In fact, in centralized systems, semantic technology has nothing to sell.
- The systems, agents, and services are heterogenous from different aspects (such as architecture, infrastructure, key components, and security mechanisms).

- SAEs are often open and widespread. Therefore, users and service requesters are often unknown and not predefined to the service providers.
- The composition of agents, services, and resources is possible.
- Interoperability and cooperation of the agents and services are widely common in these environments.

3.2 Authorization Requirements for SAEs

As the authorization model presented in this thesis is a *general* model for SAEs (that covers a broad range of semantic aware environments), we should limit ourselves to the authorization requirements that are common in all of them. It is clear that for a special SAE (e.g., Semantic Web) we may have other minor authorization requirements beside the ones listed here.

The most essential authorization requirements that should be considered in an authorization model for an SAE are, but not limited to, the following:

R1- Access policy rules should be defined in different levels of granularity and in both conceptual and ground (individual) levels. For example, for a library, we should be able to state policy rules such as "Student has read access to scientific article in the library". *Student*, *read*, and *scientific article* are concepts (or classes), which are defined in the ontologies. In fact, just having policy rules over the individuals (that are supported by traditional authorization models) is not sufficient in SAEs. For example, the statement "David can read the article X through network" is a policy rule over individuals where *David* is an individual of concept *Student*, *read through network* is an individual of concept *read*, and *article* X is an individual of concept *scientific article.*

R2- Due to the existence of semantic relationships between entities in such environments, the model needs to take semantic relationships in different domains of entities (e.g., subjects, objects, and actions) into account in policy propagation and policy inference [157, 158]. For example, if we know that $MS.Student \sqsubseteq Student$, we can propagate the permission policy rule specified in requirement R1 to MS students, and so everybody presents a credential shows that he/she is an MS student can take access to any scientific article. If we take the obligations, prohibitions, gratuities, and their conflicts into account, and also consider the semantic relationships in subjects, objects, actions, and other elements, this kind of policy propagation becomes a complex problem. Thus, it needs a reliable logical infrastructure for this purpose.

R3- The authorization model should be distributed and different authorities specify their security policies in a distributed manner [158, 157]. Therefore, such a model should be scalable as well.

R4- The model should consider detection and resolution of probable conflicts between explicit and implicit policy rules (which are inferred by semantic relationships) [96, 154, 157, 110]. For example, consider a policy rule that says "BS student cannot have access to Springer article in the library". This rule is in conflict with the permission rule specified in requirement R1, considering the subsumption relationships exist between the elements of these two policy rules. Thus, we should have a strategy to prevent or resolve such conflicts.

R5- The model should consider contextual constraints to dynamically activate policy rules based on the environmental conditions [113, 56]. In other words, the model should be context-aware.

R6- The authorization model should be abstract enough to be independent from the implementation, especially the model proposed for SAEs, where information, resources, agents, and resource providers are heterogenous [154].

R7- The requirement of specifying obligation policies beside the access policies (permissions and prohibitions) in new distributed computing environments imposes the support of obligation policy specification in the authorization model of SAEs [25, 108]. The logical relation between the obligations, permissions, and prohibitions should also be considered in adding obligations to the model. For example, it would be possible to specify that an agent is obliged to do an action, and we could derive that the agent is permitted to do the action, based on the specified obligation.

R8- Due to the distribution of the environment and the emphasis on common understanding of shared information (such as policy rules defined over the abstract layer), the model requires to have a formal clear semantics for its policy specification language. In the model that considers policy propagation and inference, semantics provide a basis for proving the soundness of the developed propagation rules in the model.

R9- Since SAEs are often open environments, the users or subjects who want to access the resources are not predefined. Thus, the authorization model is supposed to support the specification of subjects based on their attributes not only their identities in security policy rules [5, 150].

R10- It should provide some administrative facilities for easier policy management, such as exception policy specification, delegation of administration, and supporting the definition of roles and groups [142, 133, 1].

R11- Composition of services and systems, and also cooperation and interoperation through resource sharing (by creating virtual organizations) in SAEs, impose the requirement of policy composition (by different styles) in the authorization model proposing for SAEs [158, 6, 129].

Note that the requirements listed above for authorization in SAEs are not necessarily all the security requirements of these environments. On the other hand, it might be impossible to satisfy all the requirements in one model. However, these are collected based on the survey we done on the related work and the characteristics of SAEs. In this research we consider these requirements in proposing our logic-based authorization model for SAEs.

3.3 A Survey on Semantic-Aware Authorization Models

With the advent of semantic technology and developing SAEs (especially Semantic Web), challenges on providing security in these new environments was raised. Bonnati, *et al.* [31] discussed important requirements and open research issues in access control for Semantic Web. The most important issue in SAEs is the layer (in the semantic layer cake [66, 162]) in which an authorization model acts. Hence, in the following, we survey the related research based on the layer they consider.

The World Wide Web Consortium (W3C) took the flexibility of file-level access control to provide an authorization model (named ACL) on Semantic Web resources that are identified by URIs [131]. The model contains a language to specify access rules on resources. In fact, this model obeys the traditional matrix-based models and does not satisfy most of the security requirements of semantic-aware environments (e.g., having policy for unknown users, considering the semantic relationships between entities, and considering the contextual constraints and distribution of the environment).

There are some papers such as [24, 46] on providing fine-grained security (FGAC) for XML documents that construct the first layer of the semantic layer cake. However, providing security at this level cannot take the semantic relationships that are specified in higher layers in the semantic layer cake in their control procedure.

Kagal, Finin, and others proposed a policy language called Rei [96] based on Semantic Web languages like RDF and DAML+OIL and developed the Rein framework, based on this language. In Rei, concepts of deontic logic like permission, obligation, and prohibition were used. However, the Rei engine just reasons about domain-specific knowledge and not about the security policy specification [152]. Thus, it lacks modal (deontic) reasoning ability over the specified policies. Hence, modal conflict detection is performed manually in this model.

KAoS [156, 155] is a policy language accompanying with a set of services and tools, which are utilized to specify, manage, and enforce policies (defined based on deontic operators) in the environments where machines and humans, and other computing agents are interoperating. KAoS uses KPO[1] (which is described with OWL) for specification of the entities, agents, actions, and constraints. In the proposed model based on this language, only the subsumption relationships between the actions are taken into account. Also the contexts of subjects, objects, and somehow the environment are not considered.

Another policy-based approach to access control for RDF stores is proposed by Reddivari, *et al.* [134]. In their paper, a set of actions that are required to manage an RDF store is determined and an authorization model to permit or prohibit these actions is proposed. This model does not consider the relationships between the subjects and objects — which are specified in RDF-schema — in RDF triples.

Qin, *et al.*, [132] proposed a concept-level access control (CLAC) or authorization model taking into account some semantic relationships in the objects domain. The more complete model at this layer (i.e., ontology level) is our proposed authorization model, named SBAC [91, 90]. SBAC considers all the semantic relationships, which could be specified in the

[1]KAoS Policy Ontology

OWL [49] ontology language at different levels (concept, property, and individual), and in all domains of access control (subjects, objects, and actions). Both CLAC and SBAC models do not have a clear semantics for their access policies and policy inference rules, and thus lack a verification proof for the soundness and consistency of their inference rules. Also, they do not consider the obligations, contextual constraints, and distributed nature of such environments in their authorization model.

Priebe, *et al.* [130] proposed integrating users, resources, and environment attributes into semantic context. They extended the XACML architecture [120] with attributes ontologies and inference over them. In this extension, XACML is left as policy specification language. Therefore, the model cannot infer implicit policies based on the semantic relationships in the environment. The model uses ontologies to capture semantic relationships between attributes.

Semantic access control (SAC) or authorization model [162, 161] proposed by Yague, *et al.*, is a semantic model for heterogeneous and dynamic environments. SAC is a certificate-based model for applying Semantic Web layers to access control in different environments. The model includes four meta models; the PAS (Policy Applicability Specification) meta model, the Policy meta model, the SRR (Secured Resource Representation) meta model, and the SOAD (Source of Authorization Description) meta model. Semantic policy language (SPL), which is introduced in SAC, and semantic descriptions of the certificates issued by each source of authorization (SOA) are designated to validate access control policies in this model [161]. Although, the SAC model is proposed with the similar layers in the semantic layer cake and tackles the problems of distribution and multi-granularity of policies, it does not solve the main security issue of SAEs, i.e., the effect of semantic relationships on policy propagation.

Naumenko in his thesis [124] attempted to adopt Semantic Web standards for the creation of unified view on the access control area. Similar to the SAC model [161], using semantic technology in access control, enables flexible, collaborative, and distributed management of access control based on semantic relationships amongst relating concepts. This model has the same problem as the SAC model. The main disadvantage of most of the proposed models for SAEs is the lack of considering all the security requirements of SAEs, especially the ability of specifying security policies at the conceptual (ontology) level, and considering the effect of semantic relationships on propagation of different kinds of policies (e.g., obligation,

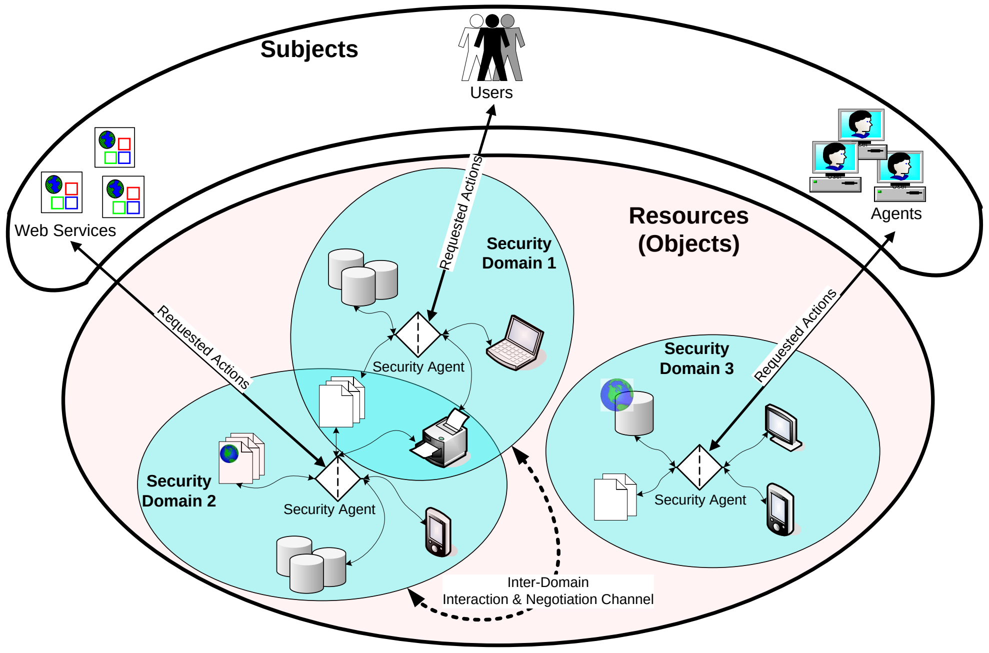


Figure 3.3: Overall authorization framework for an SAE.

permissions, and prohibitions) by a set of sound and complete inference rules. In this thesis, we propose an authorization model based on the MA(DL)^2 logic, which is proposed for policy specification and inference based on the defined semantic relationships in SAEs. In fact, MA(DL)^2 helps us to have a sound and complete basis with clear formal semantics for authorization and policy inference.

3.4 Overall Proposed Authorization Framework

Following the requirements listed for authorization models in semantic-aware environments, and the problems mentioned for the related work, we propose an overall framework and model for semantic-aware environments, their elements, and authorization in them.

3.4.1 Overall Framework of the Environment

As shown in Figure 3.3, in our framework, an SAE is divided into a number of security domains, which are not necessarily separated from each other. Each security domain (see Figure 3.4) contains the following elements:

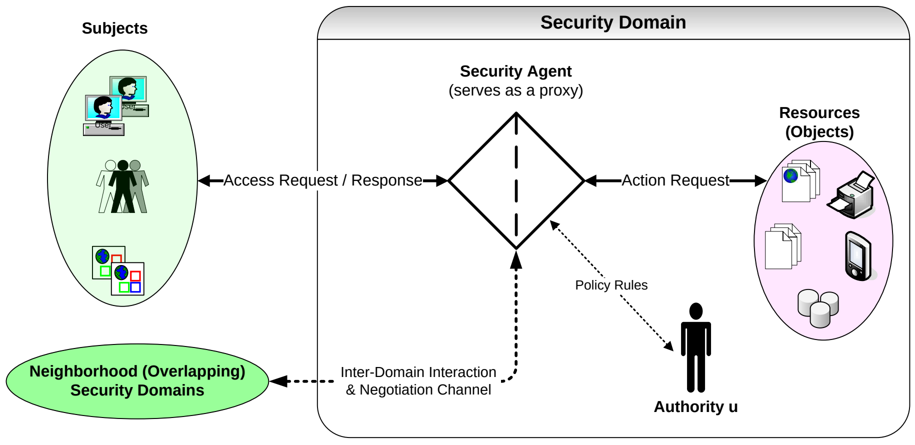


Figure 3.4: Overall framework of a security domain.

- A set of *under-protection resources*, which are registered in the security domain.
- An *authority* who specifies security policy rules (in the both conceptual and ground levels) for resources (objects) registered in the domain. The authority might be either a real primitive authority or a virtual composite authority based on the nature of the security domain.
- A *security agent* which infers and enforces the security policy rules (specified by the authority).

Due to the resource sharing in such environments and appearing shared subdomains, the important security issue in this situation is how to manage the shared resources within different security domains. A proper approach for security administration in this situation is a *cooperative management* approach, which is proposed in this thesis based on a logical infrastructure. Cooperative management, which is inherited from Management Science [21], enables us to involve all authorities of related security domains in the security decision-making process. In this approach, the policy rules of the shared or subdomains are derived from the rules stated by the participating authorities on the shared domain, based on the agreed cooperation management style. More details of this approach can be find in Section 4.3 and Chapter 6.

3.4.2 Security Agent Architecture

For implementing a security agent in a security domain, we suggest the architecture shown in Figure 3.5. The architecture is developed by adopting the existing standard security frameworks, i.e., the ITU-T access control framework [82] and the XACML framework [121]. The main components of the proposed architecture of a security agent are as follows.

PAP (Policy Administration Point) provides an interface for authorities to state security policy rules, to negotiate with other authorities to determine the cooperative management style applicable for cooperative and shared domains, and to configure the meta policy of the domain. At the request of PDP (introduced in the following), PAP may require to communicate with the other domains' PAPs (through inter-domain communication & negotiation channel) to fetch the policy rules specified over the shared resources for cooperative administration.

PDP (Policy Decision Point) decides about the access requests (using the procedure described in Section 6.3.3) and determines the obligation rules related to the access request. To this aim, PDP infers applicable security policy rules using *MA(DL)*^{2} *Inference Engine*.

PEP (Policy Enforcement Point) receives a subject's request, performs access control, and enforces applicable obligations determined by PDP. PEP contains

- *ACP* (Access Control enforcement Point), which works as a proxy of resources and controls all the access requests,
- *OEP* (Obligation Enforcement Point), which enforces the obligations related to the system or the access requester.

MA(DL)^2 **SKB** (Security Knowledge Base) contains specification of ontologies (of subjects, resources or objects, and actions), assertions about the individuals, security policy rules (SPR), and current context information (CI).

Context Handler gathers required context information from *Context Sensors* and inserts it as a set of contextual propositions into the SKB.

Credential Verifier verifies the validity of the provided credentials (in an access request) using a source of authority (SOA)[2].

[2]Readers may refer to [83] for more information about the components of Privilege Management Infrastructure (PMI).

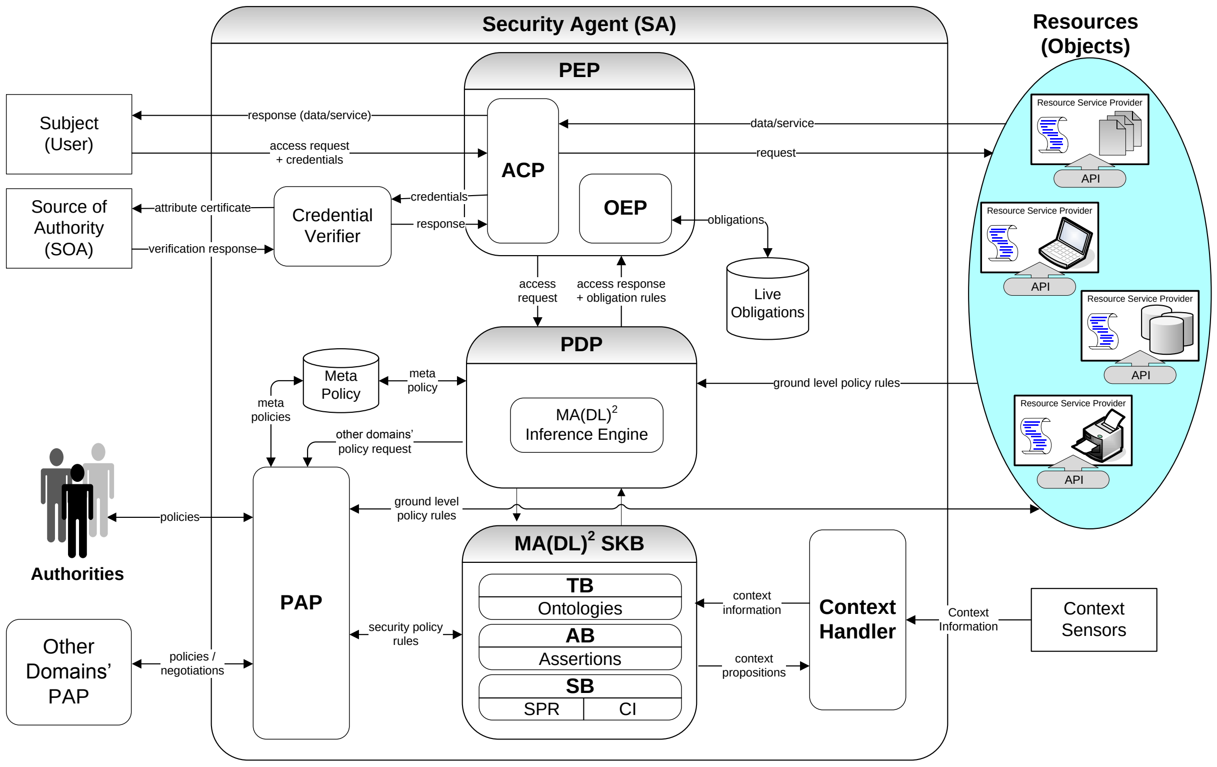


Figure 3.5: Architecture of a security agent of a security domain.

The proposed architecture enables the security agent to verify users' certificates, infer the applicable security policy rules, and enforce them. The details of access control procedure that a security agent follows is described in Section 6.3.3.

3.4.3 Canonical Resource Model

We have a canonical model for under-protection resources in a security domain. In the canonical model (shown in Figure 3.6), each resource has a *resource service provider*. The resource service provider has an standard API for receiving and replying the access requests from subjects. We leverage Web service API as an standard API in our model.

A Web service specifies a collection of services. Each possible action on a resource (object) is considered as a service in this model. Using OWL-S [50], we can specify the properties of services (actions) of a resource in the canonical model.

By taking an action as a service in the canonical model, the ground (individual) level

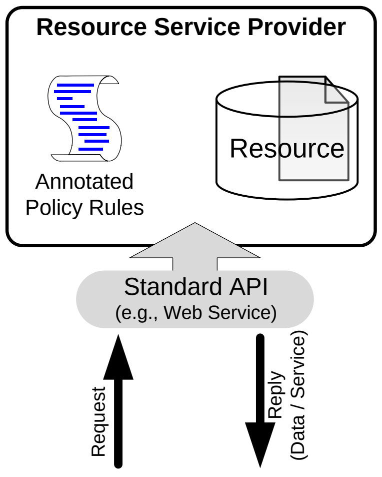


Figure 3.6: The canonical model of resources in SAEs.

policy rules related to the action on the resource are annotated in *Service Profile* (in OWL-S) of the service (see [55] for more details on policy rule annotation). Ground (individual) level policy rules are introduced in Section 6.2.2.

3.4.4 Policy Specification Logical Language

As mentioned in requirements for authorization in SAEs in Section 3.2 and considered in the proposed overall authorization framework, a proper authorization model should provide policy specification in both conceptual and ground (individual) levels. In the authorization model introduced in this thesis, a logical language, MA(DL)^2, is proposed for this purpose.

MA(DL)^2 and its properties are introduced in the next two chapters. After introducing MA(DL)^2 logical language, the formal specification of the authorization model, and the approach of using MA(DL)^2 for policy specification and inference in both conceptual and individual level is introduced in Chapter 6.

3.5 Summary

Proposing a proper authorization model for a computational environment requires to analyze its security requirements. The overall framework for authorization in SAEs, which is proposed in this chapter, is based on the collected authorization requirements for SAEs.

In this overall framework, an SAE is divided into a number of security domains. For each security domain there is a security agent with an administrator (we call it authority) and each resource (e.g., a service, a document, or a device) can register itself in one or more domains. Thus, an authority is responsible for specifying security policy rules for resources (objects) that are registered in the authority's associated domain in the conceptual and individual levels. Security agents have the duty of inferring and enforcing the security policies. The applicable policy rules are inferred based on the semantic relationships between the entities (specified in the ontologies), meta security policies determined by the authorities, and specified explicit security policy rules. Security agents act as proxy servers for resources (under-protection objects) and control all the accesses by inferring the applicable policy rules.

Policy specification and inference in this overall framework is based on the MA(DL)^2 logic family, which is introduced in the next chapter. In the rest of the thesis, after introducing the MA(DL)^2 logic family and its properties, we formally specify our proposed authorization model for SAEs based on this logic.

Chapter 4

MA(DL)^2 Logic Family

To specify security policy statements in an abstract level for semantic-aware environments, we introduce the MA(DL)^2 logic family as a combination of multi-authority (poly-modal) version of deontic logic (MADL) [67, 116] and description logic (DL) [13]; MA(DL)^2=MADL+DL. Figure 4.1 shows the members of this logic family and their relationships.

In this logic family, description logic represents semantic relationships of concepts in semantic-aware environments and multi-authority deontic logic specifies the security statements in them.

In this chapter, after a brief introduction of description logic and deontic logic, the members of MA(DL)^2 logic family are introduced in more details.

4.1 Logical Foundations of MA(DL)^2

Since the core of MA(DL)^2 is a combination of description logic and multi-authority version of standard deontic logic, a brief introduction of these two logics are represented in the rest.

4.1.1 Description Logic

A family of formal languages that are used in recent years to encode and reason about an ontology (as a knowledge base) is the *description logic (DL)* family [13]. The most important characteristic of the languages belong to this family in comparison with their precedence (such as semantic networks) is their equipment with the logic-based semantics that provides a clear foundation and common understandable knowledge about them. Another distinguished

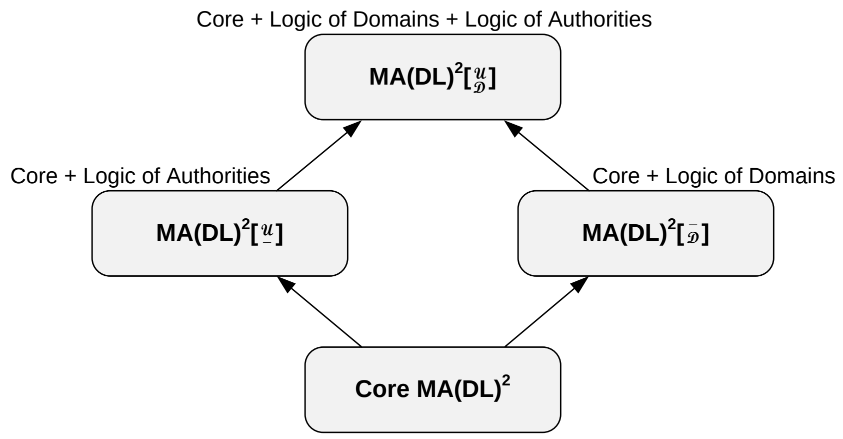

Core + Logic of Domains + Logic of Authorities
MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$
Core + Logic of Authorities
MA(DL)$^2[^{\mathcal{U}}_{-}]$
Core + Logic of Domains
MA(DL)$^2[^{-}_{\mathcal{D}}]$
Core MA(DL)^2

Figure 4.1: Members of MA(DL)^2 logic family and their relationships and expressive powers.

feature is the ability of reasoning that allows to infer implicit knowledge about concepts and properties from the explicitly represented knowledge automatically. Inference services in DL systems can help performing such inferences. In particular, *subsumption* relationships between concepts and *instance* relationships between individuals and concepts play a crucial role to this end.

There are various types of description languages that are distinguished by the constructors they provide. Most of these languages are from the family of $\mathcal{AL}$*-languages* (attributive languages) [13]. The most important constructors used in concept descriptions in this family of description languages are as follows. In fact, the following introduces the $\mathcal{ALC}$ language.

$$\begin{array}{rlr}
C, D \rightarrow & A \mid & \text{(atomic concept)} \\
& \top \mid & \text{(universal concept)} \\
& \bot \mid & \text{(bottom concept)} \\
& \neg C \mid & \text{(complement of a concept)} \\
& C \sqcap D \mid & \text{(intersection of two concepts)} \\
& C \sqcup D \mid & \text{(union of two concepts)} \\
& \forall R.C \mid & \text{(value restriction on role } R\text{)} \\
& \exists R.C \mid & \text{(existential quantification over role } R\text{)}
\end{array}$$

If we add transitive roles to $\mathcal{ALC}$, it becomes $\mathcal{ALC}_{R^+}$, which is denoted by $\mathcal{S}$. Extension of

Table 4.1: Deontic statuses and their logical definition

Notation	**Description**	**Definition** based on Obligation
`OB`	It is obligatory that	`OB`
`PE`	It is permissible that	$\texttt{PE}\,p \leftrightarrow \neg\texttt{OB}\neg p$
`IM`	It is impermissible that	$\texttt{IM}\,p \leftrightarrow \texttt{OB}\neg p$
`GR`	It is gratuitous that	$\texttt{GR}\,p \leftrightarrow \neg\texttt{OB}\,p$

$\mathcal{S}$ by inverse roles gets $\mathcal{SI}$ and by adding the hierarchy of roles, it becomes $\mathcal{SHI}$. Extension of $\mathcal{SHI}$ by functional restrictions gets $\mathcal{SHIF}$, by cardinality restrictions gets $\mathcal{SHIN}$, and by qualified number restrictions gets $\mathcal{SHIQ}$. Supporting of different data types (e.g., string, integer) and concrete domains $\mathbf{D}$ as well as nominals $\mathcal{O}$ in $\mathcal{SHIN}$, results $\mathcal{SHOIN}(\mathbf{D})$, which is a basis of OWL-DL language. OWL-DL is currently the most popular language for description of ontologies in SAEs [117].

Note that MA(DL)^2 is founded based on $\mathcal{ALC}$. Its extensions with more expressive description logics are postponed as future work.

4.1.2 Standard Deontic Logic

Deontic logic is a branch of symbolic logic that directly involves topics of practical significance such as morality, law, social, and business organizations (their norms and their normative constitution) and security systems. There are different variations of deontic logic where *standard deontic logic (SDL)* is the most cited and studied one [116].

Standard deontic logic (SDL) as a kind of normative logic is one of the first deontic logics axiomatically specified. Basically, SDL is a member of class *normal modal logics* [116]. There are four normative statuses in SDL as shown in table 4.1. All the four statuses can be defined with one of them. The most popular approach is taking `OB` as primitive and defining the rest as is shown in table 4.1. In this thesis, we use a multi-authority (or poly-modal) version of SDL forthcoming called *multi-authority deontic logic (MADL)* as a basis of our authorization model. In MADL, $\texttt{OB}_u p$ means that an authority u states that "p must be a case". The axioms and inference rules of MADL is similar to SDL, except that $\texttt{OB}\ p$ is replaced with $\texttt{OB}_u p$. The semantics of MADL is different from SDL to some extent, as is described in Section 4.2.2.

Assume that we have a language of classical propositional logic with an infinite set of propositional variables, the operators $\neg$ and $\rightarrow$, and the operator $\texttt{OB}$. Then, SDL is axiomatized as follows:

A1. If p is a tautology of propositional logic, then $\vdash p$ (TAUT)

A2. $\vdash \texttt{OB}(p \rightarrow q) \rightarrow (\texttt{OB}p \rightarrow \texttt{OB}q)$ (OB-K)

A3. $\vdash \texttt{OB}p \rightarrow \neg\texttt{OB}\neg p$ (OB-D)

R1. If $\vdash p$ and $\vdash p \rightarrow q$ then $\vdash q$ (MP)

R2. If $\vdash p$ then $\vdash \texttt{OB}p$ (OB-ONCE)

The MA(DL)^2 logic, which is proposed in this thesis, is a variant of deontic logic proposed for specification and inference of deontic norms at the conceptual level, which is obtained by combining multi-authority deontic logic (MADL) with description logic (DL).

4.2 Core MA(DL)^2 Logic

In Core MA(DL)^2, we concentrated on the effect of semantic relationships on the inference of norms defined on the concepts (at the conceptual level). As mentioned earlier, Core MA(DL)^2 is obtained by integrating description logic, multi-authority deontic logic, and n-ary predicates on concepts. In the rest of this section, Core MA(DL)^2 is introduced by its syntax, semantics, and proof theory.

4.2.1 Syntax

The description logic we embedded in MA(DL)^2 is a typed version of $\mathcal{ALC}$ [13], which is proposed in this thesis. To this aim, we introduce concept-type. Definition of the syntax and semantics of typed $\mathcal{ALC}$ are presented during the detailed introduction of MA(DL)^2 in the rest.

One of the main drawbacks of description logics is their inability in representing concrete entities (concepts). In fact, they just represent knowledge at the abstract level [111]. To overcome this constraint, *concrete domains* have been proposed by Baader and Haschke [14] and extended by others (e.g., see [111]). Similar to *concrete domains*, we define *concept-types* (call it in short *types*) in MA(DL)^2. Each concept-type represents a set of concepts describing a subset of the objects in the world. For example, in access control, a concept-type like *Sub*, represents all the concepts describing the subjects (or access requesters) in the world.

A concept-type is admissible in MA(DL)^2 if satisfies the following definition.

Definition 4.2.1 (Admissible Concept-Type) *A concept-type σ is admissible if:*

- *the concepts of type σ are closed under negation, i.e., if $C : \sigma$ then $\neg C : \sigma$, and*
- *the type σ contains a universal concept $\top_\sigma$ (and analogously $\bot_\sigma \equiv \neg\top_\sigma$).*

The advantages of having admissible concept-types are that concept negation are constrained to a concrete domain which makes it more applicable in real applications, and also it enables us to do type checking for preventing from occurring mistakes in specification, especially in security policy specification.

Following the definition of concept-types, it is clear that the set of concepts of the same type (e.g., type σ) with subsumption relation construct a lattice, where $\top_\sigma$ and $\bot_\sigma$ are its supremum and infimum elements.

Note that, since types are interpreted as sets, existence of *subtype* relation, which is interpreted as subset relation, is inevitable [137]. For the sake of simplicity, we suppose that the concept-types with subtype relation construct a lattice. Supremum and infimum types in this lattice are denoted by $\sigma_\top$ and $\sigma_\bot$ respectively, and we have $\top : \sigma_\top$ and $\bot : \sigma_\bot$, where $\top$ and $\bot$ are top and bottom concepts in regular DLs[1].

Following the introduction of concept-types, we define the alphabet of the Core MA(DL)^2 logic as follows.

Definition 4.2.2 (Alphabet of Core MA(DL)^2) *The alphabet of the Core MA(DL)^2 language includes the following elements.*

- *A finite non-empty set Σ of admissible concept-types including $\sigma_\top$ and $\sigma_\bot$.*
- *An enumerable non-empty set $\mathcal{C}$ of atomic concepts of types belong to Σ (each member is denoted by $C : \sigma \in \mathcal{C}$) including $\top_\sigma : \sigma$ (universal concept) and $\bot_\sigma : \sigma$ (bottom concept) for each concept-type $\sigma \in \Sigma$. Note that each atomic concept might be* primitive *or* defined.

[1]In fact, $\top$ and $\bot$ are aliases of $\top_{\sigma_\top}$ and $\bot_{\sigma_\bot}$ respectively.

- *an enumerable set $\mathcal{R}$ of roles of types belong to Σ^2 (e.g., $R : (\sigma_i, \sigma_j) \in \mathcal{R}$, where $\sigma_i, \sigma_j \in \Sigma$),*
- *an enumerable set $\mathcal{P}$ of predicate symbols in which each n-ary predicate symbol $p : (\sigma_1, ..., \sigma_n)$ is of a type in Σ^n (propositions are assumed as predicates with $n = 0$ and the set of them is shown by $\mathcal{P}_0$),*
- *an enumerable set $\mathcal{I}$ of individuals of types belong to Σ (each member is denoted by $a : \sigma$),*
- *a finite non-empty set of primitive authorities $\mathcal{U}$,*
- *the DL concept constructors ($\sqcap, \sqcup, \neg, \forall, \exists$),*
- *the subsumption relation (i.e., $\sqsubseteq$) and definition or specification relation (i.e., $\equiv$) for concepts,*
- *deontic statuses symbols including `OB`, `PE`, `IM`, and `GR`,*
- *the propositional primitive relaters $\wedge, \neg$, and abbreviations $\vee$, $\rightarrow$, and $\leftrightarrow$,*
- *two symbols `T` (true) and `F` (contradiction or false),*
- *auxiliary symbols '(' and ')'.*

Definition 4.2.3 (Signature) *A tuple $\mathcal{S} = \langle \Sigma, \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{I}, \mathcal{U} \rangle$ constructs a signature in the Core MA(DL)^2 language.*

In order to use the MA(DL)^2 logic in each application, it is required to define a signature at first.

In MA(DL)^2, we have three kinds of formulae, which are constructing the sentences of this language, including t-formulae, c-formulae, and a-formulae. For defining t-formulae, for the sake of simplicity, we take $\mathcal{ALC}$ description logic [13] as a basis. By typed $\mathcal{ALC}$, the set $\mathcal{C}^*$ of *complex concepts* is the smallest set that is defined as follows:

- each atomic concept $C : \sigma \in \mathcal{C}$ belongs to $\mathcal{C}^*$ as well,
- if $C : \sigma \in \mathcal{C}^*$, then $\neg C : \sigma \in \mathcal{C}^*$

- if $C_1 : \sigma_1, C_2 : \sigma_2 \in \mathcal{C}^*$, then $C_1 \sqcap C_2 : \sigma$ (resp. $C_1 \sqcup C_2 : \sigma$) belongs to $\mathcal{C}^*$ and σ is the greatest lower bound (resp. least upper bound) of types σ_1 and σ_2.
- if $C : \sigma \in \mathcal{C}^*$, $R : (\sigma_1, \sigma_2) \in \mathcal{R}$, and σ_2 is subtype of σ, then $\exists R.C : \sigma_1$ and $\forall R.C : \sigma_1$ belong to $\mathcal{C}^*$.

Definition 4.2.4 (T-Formula) *Given a signature $\mathcal{S}$, a terminological formula (in short t-formula) is defined as*

- *subsumptions like $C_1 \sqsubseteq C_2$ where $C_1 : \sigma_1 \in \mathcal{C}, C_2 : \sigma_2 \in \mathcal{C}^*$, and σ_1 is subtype of σ_2,*
- *equalities like $C_1 \equiv C_2$, where $C_1 : \sigma \in \mathcal{C}, C_2 : \sigma \in \mathcal{C}^*$, and both of them are of the same type.*

Definition 4.2.5 (A-Formula) *Given a signature $\mathcal{S}$, an assertional formula (in short a-formula) is an assertion of $C(a)$ or $R(a, b)$ where $C : \sigma_1 \in \mathcal{C}^*$ is a concept, $R : (\sigma_1, \sigma_2) \in \mathcal{R}$ is a role, $a : \sigma_1{}', b : \sigma_2{}' \in \mathcal{I}$ are individuals, $\sigma_1{}'$ is subtype of σ_1, and $\sigma_2{}'$ is subtype of σ_2.*

Definition 4.2.6 (C-Formula) *Given a signature $\mathcal{S}$, a conceptual formula (in short c-formula) is defined inductively as follows:*

- *every proposition $x_i \in \mathcal{P}_0$ is an atomic c-formula.*
- *every t-formula and every a-formula is an atomic c-formula.*
- *if $C_1 : \sigma_1, ..., C_n : \sigma_n \in \mathcal{C}^*$ are complex concepts, $p : (\sigma'_1, ..., \sigma'_n)$ is an n-ary predicate symbol, and for all i $(1 \leq i \leq n)$ σ_i is subtype of σ'_i, then $p(C_1, ..., C_n)$ is an atomic c-formula.*
- *if α is a c-formula, $u \in \mathcal{U}$ is an authority, and* `ds` *is a deontic status symbol, then $\texttt{ds}_u\alpha$ is a c-formula.*
- *if α_i and α_j are c-formulae, then so are $(\alpha_i \wedge \alpha_j)$, $\neg\alpha_i$, $(\alpha_i \vee \alpha_j)$, $(\alpha_i \rightarrow \alpha_j)$, and $(\alpha_i \leftrightarrow \alpha_j)$.*

One of the main differences between MA(DL)^2 and the other proposed hybrid languages of description logics (see e.g., [34, 54, 101, 72, 122, 117]) is its ability to define formulae and rules at the conceptual level; however, in all the proposed hybrid languages of description

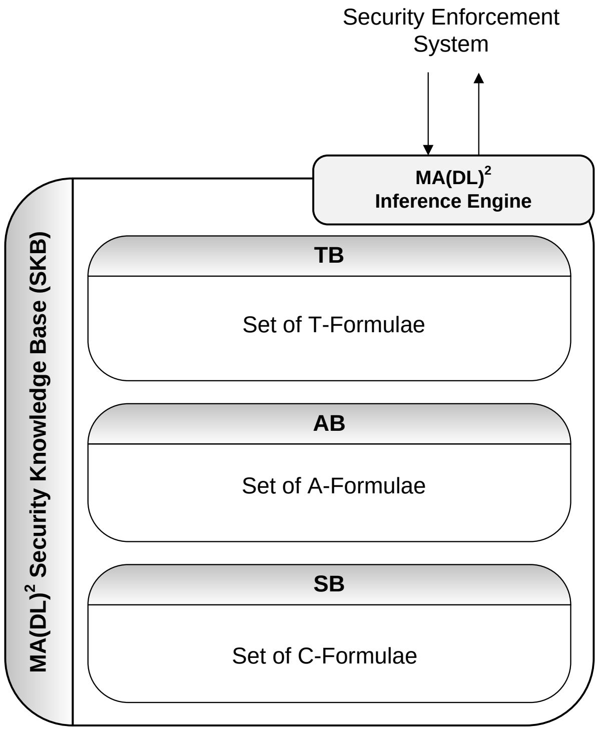

Security Enforcement System
MA(DL)^2 Inference Engine
MA(DL)^2 Security Knowledge Base (SKB)
TB
Set of T-Formulae
AB
Set of A-Formulae
SB
Set of C-Formulae

Figure 4.2: Architecture of an MA(DL)^2 security knowledge base.

logics, the rules at the individual level (in ABox) are taken into account. $\mathcal{DLR}$ [34] is another logic in which n-ary predicates are considered in a different way. n-ary predicates in $\mathcal{DLR}$ is the extension of binary roles in description logic and in fact, it is a representative of a set of n-ary relations on objects. Thus, it is different from n-ary predicates in MA(DL)^2, which are definable on specific concepts and cannot be used in concept descriptions. The meaning of n-ary predicates in MA(DL)^2 becomes more clear, when presenting its semantics.

Definition 4.2.7 (Security Knowledge Base (SKB)) *Let $\mathcal{S}$ be a signature. An MA(DL)^2 security knowledge base (SKB) is defined as $\mathcal{K} = \langle \mathcal{TB}, \mathcal{AB}, \mathcal{SB} \rangle$, where*

- *$\mathcal{TB}$ is the terminology box and includes a finite set of t-formulae in which no atomic concept is appeared more than once on the left-hand side of equalities. Also, the set of formulae is acyclic.*
- *$\mathcal{AB}$ is the assertional box and includes a finite set of a-formulae.*

- *$\mathcal{SB}$ is the security rule box and includes a finite set of c-formulae specifying a set of concept-level security policy rules and related logical formulae.*

Figure 4.2 shows the architecture of an MA(DL)^2 SKB. $\mathcal{SB}$ in SKB $\mathcal{K}$ specifies security policy rules at the abstract conceptual level. The description of complex concepts in $\mathcal{TB}$ results in inferring implicit security policy rules from the explicit ones specified in $\mathcal{SB}$. Each security policy rule in $\mathcal{SB}$ is in fact a conditional deontic rule which is defined as follows. Accompanying with the security rules in $\mathcal{SB}$, we may need to have some facts and rules (e.g., about the current contextual conditions) which are specified at the conceptual level using c-formulae.

Definition 4.2.8 (Security Policy Rule) *A security policy rule specified by an authority u is defined as a conditional deontic rule $\alpha \rightarrow \texttt{ds}_u\beta$, where $\texttt{ds}$ is a deontic status, and α and β are c-formulae.*

In a security policy rule, α is a condition of the rule to be activated. This usually describes the contextual conditions in which the policy rule is applicable. β can be defined as any required statement, e.g., it can be defined as an access predicate over the concepts of access requesters, resources (objects), and possible actions. $\texttt{ds}_u$ specifies the norms determined by authority u over the statement β.

Note that a set of security policy rules define a security policy.

4.2.2 Semantics

The semantics of MA(DL)^2 is obtained from the combination of semantics of its three components (i.e., $\mathcal{TB}, \mathcal{AB}$, and $\mathcal{SB}$). As we combined deontic logic (as a kind of modal logic) with description logic, we decided to employ the possible worlds semantics in the form of Kripke style semantics.

A Kripke-style model of MA(DL)^2 is a 5-tuple $\mathcal{M} = \langle W, \Lambda, \Delta, I, \Phi \rangle$ where:

- W is a non-empty set of possible worlds[2].

- Λ is an interpretation function that maps each authority to a binary relation on W. Each relation $\Lambda(u)$ relates a world w to world w' that are deontically ideal w.r.t w.

[2]For security purpose, in more concrete view, each possible world can be assumed as a global protection (access) state in a special context

Intuitively in world w, all the security obligations from u's security viewpoint are fulfilled and no prohibitions from u's security viewpoint have been violated. For primitive authorities, Λ is defined by $\lambda = \mathcal{U} \rightarrow 2^{(W \times W)}$. Each relation $\lambda(u)$ must be serial, i.e., $\forall w \in W : \exists w' \in W, w' \in \lambda(u)(w)$. Such a property is required for holding axiom OB-MD in the proof theory (presented in the next section). Note that in the next sections, the definition of Λ is expanded for non-primitive authorities (i.e., composite authorities).

- Δ is a non-empty enumerable set of objects in all worlds of W. In a semantic-aware environment, we assume that all the possible worlds share the same domain of objects. In other words we have the *constant domain assumption* [15].
- I is an interpretation function that in each world w assigns
  - to each type σ, a subset of Δ, i.e., $[\![\sigma]\!]^I_w = \Delta_\sigma \subseteq \Delta$ [3],
  - to every concept $C : \sigma$, a subset of Δ_σ, i.e. $[\![C]\!]^I_w \subseteq \Delta_\sigma$,
  - to every role $R : (\sigma_i, \sigma_j)$, a binary relation on $\Delta_{\sigma_i} \times \Delta_{\sigma_j}$, i.e., $[\![R]\!]^I_w \subseteq \Delta_{\sigma_i} \times \Delta_{\sigma_j}$,
  - to every n-ary predicate $p : (\sigma_1, ..., \sigma_n)$, an n-ary relation on $\Delta_{\sigma_1} \times ... \times \Delta_{\sigma_n}$, i.e. $[\![p]\!]^I_w \subseteq \Delta_{\sigma_1} \times ... \times \Delta_{\sigma_n}$,
  - to every individual $a : \sigma$, an object in Δ_σ, i.e., $[\![a]\!]^I_w \in \Delta_\sigma$. Note that in this logic we have *unique name assumption (UNA)*[4] [13] for objects.
  - to universal concept ($\top_\sigma$) and bottom concept ($\bot_\sigma$) of type σ, the set Δ_σ and $\varnothing$ respectively, i.e., $[\![\top_\sigma]\!]^I_w = \Delta_\sigma$, $[\![\bot_\sigma]\!]^I_w = \varnothing$.

The interpretation of complex concepts are defined inductively as follows:

$$
\begin{aligned}
&[\![\neg C]\!]^I_w = \Delta_\sigma \setminus [\![C]\!]^I_w \\
&[\![C_i \sqcap C_j]\!]^I_w = [\![C_i]\!]^I_w \cap [\![C_j]\!]^I_w \\
&[\![C_i \sqcup C_j]\!]^I_w = [\![C_i]\!]^I_w \cup [\![C_j]\!]^I_w \\
&[\![\exists R.C]\!]^I_w = \{a | \exists b, \langle a, b\rangle \in [\![R]\!]^I_w \text{ and } b \in [\![C]\!]^I_w\} \\
&[\![\forall R.C]\!]^I_w = \{a | \forall b, \text{ if } \langle a, b\rangle \in [\![R]\!]^I_w \text{ then } b \in [\![C]\!]^I_w\}
\end{aligned}
$$

[3]If σ be subtype of σ', then $\Delta_\sigma \subseteq \Delta_{\sigma'}$. For more details on subtypes specification see [137].

[4]UNA means if we have two individuals a and b, then $[\![a]\!]^I_w \neq [\![b]\!]^I_w$. This assumption is required in some of the proofs presented in the next chapter.

Function I must satisfy the limitation that the interpretation of concepts, roles, and individuals, are identical in all worlds due to the philosophy of using ontologies in distributed semantic-aware environments: , i.e., for all $w, w' \in W$, $[\![C]\!]^I_w = [\![C]\!]^I_{w'}$, $[\![R]\!]^I_w = [\![R]\!]^I_{w'}$, and $[\![a]\!]^I_w = [\![a]\!]^I_{w'}$.

- Φ is an interpretation function that maps each formula related to each component of an MA(DL)^2 security knowledge base to a subset of possible worlds where the formula is correct.

  The function for t-formulae (terminological axioms) is defined as follows:

$$\Phi(C \sqsubseteq C') = \{w|\ [\![C]\!]^I_w \subseteq [\![C']\!]^I_w\}$$
$$\Phi(C \equiv C') = \{w|\ [\![C]\!]^I_w = [\![C']\!]^I_w\}$$

  Since for all $w, w' \in W$, we have $[\![C]\!]^I_w = [\![C]\!]^I w'$, we can define Φ for t-formulae as follows.

$$\Phi(C \sqsubseteq C') = \begin{cases} W, \text{if } \forall w \in W, [\![C]\!]^I_w \subseteq [\![C']\!]^I_w \\ \varnothing, \text{otherwise} \end{cases}$$

$$\Phi(C \equiv C') = \begin{cases} W, \text{if } \forall w \in W, [\![C]\!]^I_w = [\![C']\!]^I_w \\ \varnothing, \text{otherwise} \end{cases}$$

  For atomic a-formulae, Φ is defined as follows:

$$\Phi(C(a)) = \{w|[\![a]\!]^I_w \in [\![C]\!]^I_w\}$$
$$\Phi(R(a,b)) = \{w|\langle [\![a]\!]^I_w, [\![b]\!]^I_w\rangle \in [\![R]\!]^I_w\}$$

  Since for all $w, w' \in W$, we have $[\![a]\!]^I_w = [\![a]\!]^I w'$ and $[\![R]\!]^I_w = [\![R]\!]^I w'$, we can define Φ for a-formulae as follows.

$$\Phi(C(a)) = \begin{cases} W, \text{if } \forall w \in W, [\![a]\!]^I_w \in [\![C]\!]^I_w \\ \varnothing, \text{otherwise} \end{cases}$$

$$\Phi(R(a,b)) = \begin{cases} W, \text{if } \forall w \in W, \langle [\![a]\!]^I_w, [\![b]\!]^I_w\rangle \in [\![R]\!]^I_w \\ \varnothing, \text{otherwise} \end{cases}$$

The function for c-formulae is defined inductively as follows:

$$\begin{aligned}
&\Phi(x_i) \subseteq W \quad \text{,if } x_i \text{ is a proposition}\\
&\Phi(p(C_1, ..., C_n)) = \{w | \prod_{i=1}^{n} [\![C_i]\!]^I_w \subseteq [\![p]\!]^I_w\}\\
&\Phi(\neg\beta) = W \setminus \Phi(\beta)\\
&\Phi(\beta \wedge \beta') = \Phi(\beta) \cap \Phi(\beta')\\
&\Phi(\mathtt{OB}_u\beta) = \{w | \Lambda(u)(w) \subseteq \Phi(\beta)\}\\
&\Phi(\mathtt{IM}_u\beta) = \{w | \Lambda(u)(w) \subseteq \Phi(\neg\beta)\}\\
&\Phi(\mathtt{PE}_u\beta) = \{w | \Lambda(u)(w) \cap \Phi(\beta) \neq \varnothing\}\\
&\Phi(\mathtt{GR}_u\beta) = \{w | \Lambda(u)(w) \cap \Phi(\neg\beta) \neq \varnothing\}
\end{aligned}$$

Note that $\Phi(\mathcal{K})$ for a security knowledge base $\mathcal{K}$ is equivalent to the intersection of $\Phi(\alpha_i)$'s where α_is are formulae of the components of $\mathcal{K}$.

Definition 4.2.9 (Truth) *A formula α in a model $\mathcal{M} = \langle W, \Lambda, \Phi, \Delta, I\rangle$ at a world $w \in W$ is true, denoted $\vDash^{\mathcal{M}}_w \alpha$, if and only if $w \in \Phi(\alpha)$. Analogously, α at a world w is not true, denoted $\nvDash^{\mathcal{M}}_w \alpha$ if and only if $w \notin \Phi(\alpha)$.*

Definition 4.2.10 (Satisfiability and Validity) *A formula α is satisfiable, if and only if there exists a model $\mathcal{M}$ in which there exists a world w, such that $\vDash^{\mathcal{M}}_w$. A model $\mathcal{M}$* satisfies *a formula α, denoted $\vDash^{\mathcal{M}} \alpha$, if and only if $\vDash^{\mathcal{M}}_w \alpha$ for all $w \in W$. Analogously, a formula α is* valid *denoted $\vDash \alpha$ if and only if every model like $\mathcal{M}$ satisfies the formula α; in other words $\vDash^{\mathcal{M}} \alpha$ for all models $\mathcal{M}$.*

Regarding the above definition of *truth* and the Kripke model of the Core MA(DL)^2 logic the following definition is obtained.

Definition 4.2.11 *Let $\mathcal{M}$ be a Kripke model of Core MA(DL)^2 and Γ be a set of formulae in Core MA(DL)^2. Then, we have $\Gamma \vDash^{\mathcal{M}} \alpha$ iff $\Phi(\Gamma) \subseteq \Phi(\alpha)$. Also, $\Gamma \vDash \alpha$ if and only if for every model $\mathcal{M}$, if $\vDash^{\mathcal{M}} \Gamma$, then $\vDash^{\mathcal{M}} \alpha$.*

Proposition 4.2.1 *Let $\mathcal{M} = \langle W, \Lambda, \Phi, \Delta, I\rangle$ be a Kripke model of Core MA(DL)^2 and $w \in W$. We have:*

(i) $\models^{\mathcal{M}}_{w} p \to q$ *iff if* $\models^{\mathcal{M}}_{w} p$*, then* $\models^{\mathcal{M}}_{w} q$

(ii) $\models^{\mathcal{M}}_{w} \neg p$ *iff* $\not\models^{\mathcal{M}}_{w} p$

(iii) $\models^{\mathcal{M}}_{w} p \wedge q$ *iff* $\models^{\mathcal{M}}_{w} p$ *and* $\models^{\mathcal{M}}_{w} q$

(iv) $\models^{\mathcal{M}}_{w} \mathtt{OB}_u p$ *iff for all* $w' \in \Lambda(u)(w)$*, we have* $\models^{\mathcal{M}}_{w'} p$

(v) $\models^{\mathcal{M}}_{w} C \sqsubseteq C'$ *iff for all* $w' \in W$*, we have* $\models^{\mathcal{M}}_{w'} C \sqsubseteq C'$

Proof. It is straight-forward. ■

4.2.3 Proof Theory

The Hilbert style proof theory of Core MA(DL)^2 is presented as a set of axioms and inference rules as follows.

A1. If p is a truth-functional tautology over our presented language, then $\vdash p$ (TAUT)

A2. $\vdash \mathtt{OB}_u(p \to q) \to (\mathtt{OB}_u p \to \mathtt{OB}_u q)$ (OB-MK)

A3. $\vdash \mathtt{OB}_u p \to \neg \mathtt{OB}_u \neg p$ (OB-MD)

A4. $\vdash C \sqsubseteq C' \to \mathtt{OB}_u(C \sqsubseteq C')$ (SOB)

A5. $\vdash C(a) \to \mathtt{OB}_u(C(a))$ (COB)

A6. $\vdash R(a,b) \to \mathtt{OB}_u(R(a,b))$ (ROB)

A7. $\vdash C_i{}' \sqsubseteq C_i \to (p(C_1, ..., C_i, ..., C_n) \to p(C_1, ..., C_i{}', ..., C_n))$ (SPP)

A8. $\vdash p(C_1, ..., C_i, ..., C_n) \wedge p(C_1, ..., C_i{}', ..., C_n) \to p(C_1, ..., C_i \sqcup C_i{}', ..., C_n)$ (UPP)

A9. $\vdash p(..., \bot, ...)$ (BP)

A10. $\vdash C \equiv C' \leftrightarrow C \sqsubseteq C' \wedge C' \sqsubseteq C$ (EQ-Def)

A11. $\vdash \mathtt{PE}_u p \leftrightarrow \neg \mathtt{OB}_u \neg p$ (PE-Def)

A12. $\vdash \mathtt{IM}_u p \leftrightarrow \mathtt{OB}_u \neg p$ (IM-Def)

A13. $\vdash \mathtt{GR}_u p \leftrightarrow \neg \mathtt{OB}_u p$ ($\mathtt{GR}$-Def)

A14. All axioms obtained by adaptation of inference rules introduced in [136] for subsumption inference in $\mathcal{ALC}$. (DLC)

- $\vdash C_1 \sqcap C_2 \sqsubseteq C_1$
- $\vdash (C_1 \sqsubseteq C_2) \wedge (C_1 \sqsubseteq C_3) \rightarrow (C_1 \sqsubseteq C_2 \sqcap C_3)$
- $\vdash (C_1 \sqsubseteq C_2) \wedge (C_2 \sqcap C_3 \sqsubseteq C_4) \rightarrow (C_1 \sqcap C_3 \sqsubseteq C_4)$
- $\vdash \bot_\sigma \sqsubseteq C$
- $\vdash C \sqsubseteq \top_\sigma$
- $\vdash (C_1 \sqsubseteq C_2) \leftrightarrow (C_1 \sqcap \top_\sigma \sqsubseteq C_2)$
- $\vdash (C_2 \sqsubseteq C_1) \wedge (C_3 \sqsubseteq C_1) \rightarrow (C_2 \sqcup C_3 \sqsubseteq C_1)$
- $\vdash (\top_\sigma \sqsubseteq C) \leftrightarrow (\top_\sigma \sqsubseteq \forall R.C)$
- $\vdash (C \sqsubseteq \bot_\sigma) \leftrightarrow (\exists R.C \sqsubseteq \bot_\sigma)$

A15. All axioms obtained by adaptation of inference rules introduced in [136] for instance checking in $\mathcal{ALC}$. (DLA)

- $\vdash C_2(a) \wedge C_2 \sqsubseteq C_1 \rightarrow C_1(a)$
- $\vdash \forall R.C(a_1) \wedge R(a_1, a_2) \rightarrow C(a_2)$
- $\vdash R(a_1, a_2) \wedge C(a_2) \rightarrow \exists R.C(a_1)$
- $\vdash C_1(a) \wedge C_2(a) \rightarrow C_1 \sqcap C_2(a)$

R1. If $\vdash p$ and $\vdash p \rightarrow q$, then $\vdash q$ (MP)

R2. If $\vdash p$ then $\vdash \mathtt{OB}_u p$ ($\mathtt{OB}$-MO)

Axioms TAUT, OB-MK, and OB-MD as well as inference rules MP and Ob-MO are donated from multi-authority (poly-modal) version of deontic logic. Axioms PE-Def, IM-Def, and GR-Def define deontic statuses other than $\mathtt{OB}$.

SOB, COB, ROB, SPP, UPP, and BP are dedicated axioms of Core MA(DL)^2 that do not exist in description logic and deontic logic. Axiom SOB says that the description of concepts in the abstract layer in SAEs, are necessarily acceptable by all authorities. This means there is a shared conceptualization in SAEs. Axioms COB and ROB have the analogous philosophy of existence. Axiom SPP is one of the important axioms in our application (i.e., authorization and policy inference). SPP shows the impact of subsumption relationships on the propagation of n-ary predicates defined on the concepts. Axiom UPP complements axiom SPP and considers the impact of operator $\sqcup$ in description of concepts. It is obvious that ech property holds on two concepts, holds on their union, too. Axiom BP says every property (predicate) holds for bottom concept. This axiom is not useful in practice; however, it is required to have a complete proof theory.

Axioms DLC, DLA, and EQ-Def are from $\mathcal{ALC}$. Note that following the existence of axioms SOB and SPP, inference of subsumption relationships in descriptions of $\mathcal{TB}$ can be done by every possible inference system[5]. The axioms presented in this section for this purpose (in DLC and DLA) help to complete the presented Hilbert style proof theory of Core MA(DL)^2.

Following the above proof theory, some useful theorems are as follows:

- $\vdash p(C_1, ..., C_i \sqcup C_i', ..., C_n) \rightarrow p(C_1, ..., C_i, ..., C_n)$
- $\vdash p(C_1, ..., C_i, ..., C_n) \rightarrow p(C_1, ..., C_i \sqcap C_i', ..., C_n)$
- $\vdash p(C_1, ..., C_i \sqcap C_i', ..., C_n) \wedge p(C_1, ..., \neg C_i, ..., C_n) \rightarrow p(C_1, ..., C_i', ..., C_n)$, since $\neg C_i' \sqsubseteq (C_i \sqcap C_i') \sqcup \neg C_i'$, it is immediately concluded from axiom UPP.
- $\vdash C_i \equiv C_i' \rightarrow (p(C_1, ..., C_i, ..., C_n) \leftrightarrow p(C_1, ..., C_i', ..., C_n))$

Definition 4.2.12 (Provability) *The set of* provable *formulae in MA(DL)^2 is the smallest set Γ such that*

- *each instance of every introduced axiom schema are in Γ,*
- *Γ is closed under the introduced inference rules.*

Each member α of provable set Γ is a provable formula *and is denoted by $\vdash \alpha$.*

Definition 4.2.13 (Consistency) *A set Γ of formulae in MA(DL)^2 is* inconsistent *iff there are $\alpha_1, ..., \alpha_n \in \Gamma$ such that $\vdash (\alpha_1 \wedge ... \wedge \alpha_n) \rightarrow F$; otherwise Γ is* consistent*.*

By the definition of consistency, we say that a formula α is *derivable from* a set Γ of formulae in MA(DL)^2 (denoted by $\Gamma \vdash \alpha$) iff $\Gamma \cup \{\neg\alpha\}$ is inconsistent. Obviously, $\vdash \alpha$ iff $\varnothing \vdash \alpha$.

[5]Similar to axiom TAUT where we do not force any inference system for inferring the tautologies.

4.3 MA(DL)$^2[^{\mathcal{U}}_{-}]$ Logic

In SAEs, as multi-security-domain (MSD) environments, due to resource sharing for collaborative activities, and the possibility of policy specification by different authorities for the shared subdomains, cooperative administration is a fundamental security requirement. Pearlman, *et al.* [129] after introducing virtual organizations (VO) and virtual communities in which collaborative activities are made possible through resource sharing among multiple institutions, they address policy specification and enforcement in VOs as a key problem in these environments.

Extending Core MA(DL)^2 with the logic of composite authorities, enables the authorities to state their policies for the shared domains independently, and then the security agents can infer applicable policy rules of the shared domains from the policy rules specified by different authorities at that domains. This logic provides three styles of cooperative administration, namely collaborative, disjunctive, and delegative.

The technique used in this research for cooperative security administration is based on the concept of composite authority. A composite authority is a virtual authority that is a representative of primitive authorities participating in the cooperative administration based on one of the aforementioned cooperative administration styles. Participating authorities can agree upon the cooperative administration style.

4.3.1 Syntax

Augmenting MA(DL)^2 by logic of composite authorities results in MA(DL)$^2[^{\mathcal{U}}_{-}]$. The definition and syntax of composite authorities is as follows.

Definition 4.3.1 (Composite Authority) *Given a set $\mathcal{U}$ of primitive authorities, and $\&, \triangleright, |$ notations, a composite authority is defined inductively as follows:*

- *each primitive authority like $u \in \mathcal{U}$ is a composite authority.*
- *if u_i and u_j are composite authorities, then so are $(u_i \& u_j)$, $(u_i | u_j)$, and $(u_i \triangleright u_j)$.*

The smallest set of composite authorities obtained from set $\mathcal{U}$ of primitive authorities is denoted by $\mathcal{U}^*$.

Following the definition of composite authorities, we introduce different cooperative management styles (resulted from different kinds of composite authorities administrating the shared subdomains) for multi-domain environments such as SAEs.

Disjunctive Administration

Two authorities may enter into an agreement for disjunctive security administration of a shared domain. In disjunctive administration, the system grants an access if at least one of the participating authorities authorizes the access; however, it enforces an obligation or denies an access, if both participating authorities oblige it. In fact, the disjunctive composition of two authorities is less strict than the two.

In order to support the disjunctive administration style, a composite authority $(u_i|u_j)$ (read u_i disjoint u_j) is added to our calculus. The disjunctive administration style is defined by the fact that the norms of the disjoint composite authority $(u_i|u_j)$ is the union of the participating authorities' norms. This is obtained through the union of the worlds relations of the participating authorities in the semantics.

Delegative Administration

Delegation means assigning a part or the whole of someone's privileges to someone else. In information security and especially access control, three types of delegation can be identified:

1. Administration Delegation: an authority delegates his/ her administration privileges of making security statements to other authorities. Hence, the delegated authority can quote security statements from the delegator authority. As an example of this kind of delegation see [2].

2. Access Delegation: a subject (or access requester) delegates his/her rights of accessing resources (objects) to other subjects. In this way the delegated subject can obtain access to the resources that are privileged to the delegator subject. Abadi *et al.,* in [4] introduced this type of delegation formally and it has been used in other researches like [104].

3. Decision (Policy Enforcement) Delegation: an authorizer or a system that enforces policies or makes decisions on accessing subjects to the resources, delegates its policy enforcement or decision making privilege to another authorizer. As a sample, you

can see the delegation of policy execution in policy-based systems in [74]. Furthermore, this type of delegation is suitable to be used in privacy control, which is proposed in [163].

In this thesis, by *delegation* we mean *administration delegation.* By the definition of delegation, delegative administration means inferring and enforcing security policies by an authority on behalf of another authority. In this way, $(u_i \triangleright u_j)$ denotes that u_i on behalf of u_j privileges can enact statements. For example, $\mathtt{OB}_{(u_i \triangleright u_j)}\alpha$ means u_i states on behalf of u_j that α ought to be the case.

It is worthwhile to note that an authority needs to have a privilege for specifying statements on behalf of another authority. Using logic allows us to abstract away from these implementation details. However, we suppose that trust infrastructures like PKI [83] or the delegation network that is proposed in [2] handles delegation details.

Collaborative Administration

Collaborative or joint administration is a stricter approach in comparison with the disjunctive administration style. In this style, each authority can make a positive or negative obligation statement instead of all the participating authorities. However, we can not easily extend this rule to authorization statements like permission or gratuitous statements. The rest of this section clarifies this more precisely by giving the semantics of the collaborative administration calculus.

A composite authority, $(u_i \& u_j)$ (read it u_i joint u_j) denotes that u_i and u_j enact collaboratively for their shared domain.

Hybrid Administration

We may require the application of different combinations of the three aforementioned administration styles in security management of cooperative domains. For example, $((u_1 \triangleright u_2) \& (u_3 \triangleright u_4))$ is a composite authority represents that u_1 on behalf of u_2 jointly with u_3 on behalf of u_4 administrate a shared domain. To this end, complicated compositions of authorities are supported by the proposed logic.

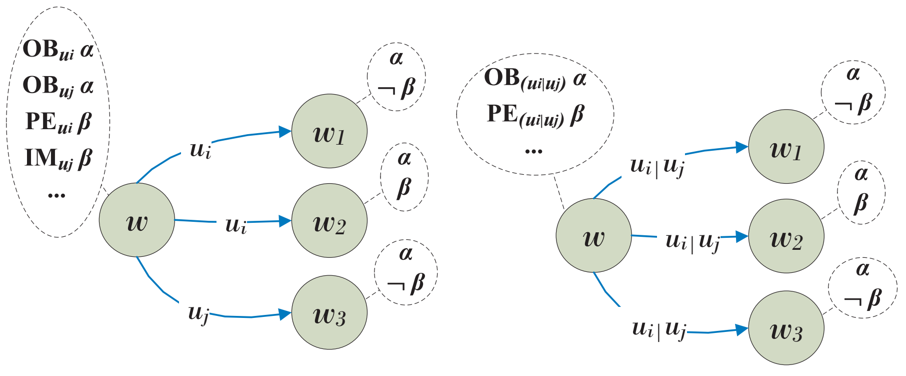


Figure 4.3: Semantics of disjunctive composition of authorities in MA(DL)$^2[^{\mathcal{U}}_{-}]$.

4.3.2 Semantics

The semantics of MA(DL)$^2[^{\mathcal{U}}_{-}]$ is obtained by refining the definition of Λ in Kripke structure of MA(DL)^2 as follows. This function for primitive authorities is defined as $\lambda = \mathcal{U} \rightarrow 2^{(W\times W)}$ and for composite authorities is extended as follows.

$$\Lambda(u) = \lambda(u), \text{if } u \in \mathcal{U}$$
$$\Lambda(u_i|u_j) = \Lambda(u_i) \cup \Lambda(u_j)$$
$$\Lambda(u_i \& u_j) = \Lambda(u_i) \cap \Lambda(u_j)$$
$$\Lambda(u_i \triangleright u_j) = \Lambda(u_i) \circ \Lambda(u_j)$$

The union of relations $\Lambda(u_i)$ and $\Lambda(u_j)$ results in the union of the norms of u_i and u_j, and intersection of these relations results in the intersection of the norms of u_i and u_j. It is clear that in the bigger (smaller) set of norms, we derive the more (less) authorizations and less (more) obligations. Figure 4.3 shows this for disjunctive composition more precisely.

The norms of composite authority $(u_i \triangleright u_j)$ is the the norms of u_j from u_i's viewpoint. Thus, it is obtained by composition of relations $\Lambda(u_i)$ with $\Lambda(u_j)$. Figure 4.4 shows the semantics of delegative authority visually.

4.3.3 Proof Theory

The proof theory of MA(DL)$^2[^{\mathcal{U}}_{-}]$ is resulted from the combination of proof theory mentioned for Core MA(DL)^2 and the axioms presented in the rest for logic of authorities. These axioms

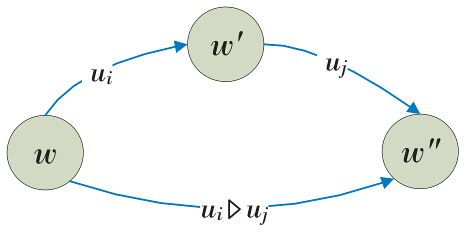


Figure 4.4: Semantics of delegative composition of authorities in MA(DL)$^2[^{\mathcal{U}}_{-}]$.

are presented in three groups, based on the three types of composite authorities.

Axioms of Disjunctive Composition

The only axiom of disjunctive composition (for disjunctive administration style) is as follows.

- $\vdash \mathtt{OB}_{(u_i|u_j)}\alpha \leftrightarrow \mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{u_j}\alpha$ (JAD)

Axiom JAD states that whenever two authorities make an obligation statement disjunctively, it means that both of them have the same obligation statement separately and they have agreed upon.

Regarding the above axiom, the following theorems hold. JAI, JAC, and JAA show that the disjoint operator ($|$) is idempotent, commutative, and associative over authorities.

- $\vdash \mathtt{OB}_{(u_i|u_i)}\alpha \leftrightarrow \mathtt{OB}_{u_i}\alpha$ (JAI)
- $\vdash \mathtt{OB}_{(u_i|u_j)}\alpha \leftrightarrow \mathtt{OB}_{(u_j|u_i)}\alpha$ (JAC)
- $\vdash \mathtt{OB}_{(u_i|(u_j|u_k))}\alpha \leftrightarrow \mathtt{OB}_{((u_i|u_j)|u_k)}\alpha$ (JAA)

By the definitions of modal statuses and axiom JAD, we get the following useful theorems.

- $\vdash \mathtt{PE}_{(u_i|u_j)}\alpha \leftrightarrow \mathtt{PE}_{u_i}\alpha \vee \mathtt{PE}_{u_j}\alpha$
- $\vdash \mathtt{IM}_{(u_i|u_j)}\alpha \leftrightarrow \mathtt{IM}_{u_i}\alpha \wedge \mathtt{IM}_{u_j}\alpha$
- $\vdash \mathtt{GR}_{(u_i|u_j)}\alpha \leftrightarrow \mathtt{GR}_{u_i}\alpha \vee \mathtt{GR}_{u_j}\alpha$

The above theorems show that for deriving obligation policies in disjunctive administration style (by `OB` or `IM` statements), we require to have the both authorities' words over. However, for deriving authorization policies (by `PE` or `GR` statements), having one of the authorities' word is enough.

Axioms of Delegative Composition

The required axioms of delegative composition (for delegative administration style) are as follows.

- $\vdash \texttt{OB}_{(u_i \triangleright u_j)}\alpha \leftrightarrow \texttt{OB}_{u_i}(\texttt{OB}_{u_j}\alpha)$ (DAD)
- $\vdash \texttt{OB}_{(u_i \triangleright u_i)}\alpha \rightarrow \texttt{OB}_{u_i}\alpha$ (DAI)

Axiom DAD represents that in the delegative administration, the case α is obligatory from the $(u_i \triangleright u_j)$ point of view, if and only if in any state, u_i accepts that α is obligatory from u_j's point of view. Axiom DAI imposes that the delegation operator ($\triangleright$) is idempotent. Note that the reverse of DAI also holds, which is redundant (we also refer to it as DAI).

By the above axioms, we get the following theorem that shows the delegation operator is associative.

- $\vdash \texttt{OB}_{(u_i \triangleright (u_j \triangleright u_k))}\alpha \leftrightarrow \texttt{OB}_{((u_i \triangleright u_j) \triangleright u_k)}\alpha$ (DAA)

Regarding the DAD axiom the following theorems hold for other statuses than $\texttt{OB}$.

- $\vdash \texttt{PE}_{(u_i \triangleright u_j)}\alpha \leftrightarrow \texttt{PE}_{u_i}(\texttt{PE}_{u_j}\alpha)$
- $\vdash \texttt{IM}_{(u_i \triangleright u_j)}\alpha \leftrightarrow \texttt{OB}_{u_i}(\texttt{IM}_{u_j}\alpha)$
- $\vdash \texttt{GR}_{(u_i \triangleright u_j)}\alpha \leftrightarrow \texttt{PE}_{u_i}(\texttt{GR}_{u_j}\alpha)$

The above results reveal that we can infer that a case α is permissible or gratuitous from the viewpoint of u_i on behalf of u_j, if the permissible or gratuitous statement of u_j is permissible from u_i's point of view. However, for obligatory or impermissible cases, we are stricter and require an obligatory statement of u_i over the obligation or impermissible statement of u_j.

Axioms of Collaborative Composition

The axioms of collaborative composition (for collaborative administration style) are as follows.

- $\vdash \texttt{OB}_{u_i}\alpha \vee \texttt{OB}_{u_j}\alpha \rightarrow \texttt{OB}_{u_i \& u_j}\alpha$ (CAD)
- $\vdash \texttt{OB}_{(u_i \& u_i)}\alpha \rightarrow \texttt{OB}_{u_i}\alpha$ (CAI)

- $\vdash \texttt{OB}_{u_i \& u_j}\alpha \rightarrow \texttt{OB}_{u_j \& u_i}\alpha$ (CAC)
- $\vdash \texttt{OB}_{u_i \& (u_j \& u_k)}\alpha \leftrightarrow \texttt{OB}_{(u_i \& u_j) \& u_k}\alpha$ (CAA)

In collaborative administration, an obligation statement is derived when at least one of the participated authorities enact the obligation statement. This principle appears as the CAD axiom in the proposed calculus. Idempotency, commutativity, and associativity of joint operator (denoted by &) are added by axioms CAI, CAC, and CAA respectively. Note that the reverse of axioms CAI and CAC also hold, which can be derived from the aforementioned axioms and thus are redundant.

By the definitions of modal statuses, and the OB-MD and CAD axioms we get the following theorems.

- $\vdash \texttt{PE}_{(u_i \& u_j)}\alpha \rightarrow \texttt{PE}_{u_i}\alpha \wedge \texttt{PE}_{u_j}\alpha$
- $\vdash \texttt{IM}_{u_i}\alpha \vee \texttt{IM}_{u_j}\alpha \rightarrow \texttt{IM}_{(u_i \& u_j)}\alpha$
- $\vdash \texttt{GR}_{u_i \& u_j}\alpha \rightarrow \texttt{GR}_{u_i}\alpha \wedge \texttt{GR}_{u_j}\alpha$
- $\vdash \texttt{OB}_{u_i \& u_j}\alpha \rightarrow \texttt{PE}_{u_i}\alpha \wedge \texttt{PE}_{u_j}\alpha$
- $\vdash \texttt{IM}_{u_i \& u_j}\alpha \rightarrow \texttt{GR}_{u_i}\alpha \wedge \texttt{GR}_{u_j}\alpha$

Axioms of Hybrid Composition

The essential axioms for having hybrid administration are as follows.

- $\vdash \texttt{OB}_{(u_i \& (u_j | u_k))}\alpha \leftrightarrow \texttt{OB}_{(u_i \& u_j) | (u_i \& u_k)}\alpha$ (DCJ)
- $\vdash \texttt{OB}_{u_i \triangleright (u_j \& u_k)}\alpha \rightarrow \texttt{OB}_{(u_i \triangleright u_j) \& (u_i \triangleright u_k)}\alpha$ (DDC)

Axiom DCJ shows the left-distribution of the collaboration operator (&) over the disjunction operator (|). Axiom DDC shows the left-distribution of the delegation operator ($\triangleright$) over the collaboration operator (&); although the reverse does not hold. The similar formulae for right-distribution also hold, which are redundant.

The distribution of disjunction operator (|) over collaboration operator (&) from left and right, and the distribution of the delegation operator ($\triangleright$) over the disjunction operator (|) also hold as the following theorems.

- $\vdash \mathtt{OB}_{u_i|(u_j\&u_k)}\alpha \leftrightarrow \mathtt{OB}_{(u_i|u_j)\&(u_i|u_k)}\alpha$ (DJC)

- $\vdash \mathtt{OB}_{u_i\rhd(u_j|u_k)}\alpha \leftrightarrow \mathtt{OB}_{(u_i\rhd u_j)|(u_i\rhd u_k)}\alpha$ (DDJ)

Proof. DDJ can be obtained easily from axioms DAD and JAD. DJC is proven to hold as follows.

[If Part] By distributing & over | using axiom DCJ in two steps and using the CAI and JAD axioms we have

$$\mathtt{OB}_{(u_i|u_j)\&(u_i|u_k)}\alpha \leftrightarrow \mathtt{OB}_{((u_i|u_j)\&u_i)|((u_i|u_j)\&u_k)}\alpha \leftrightarrow \mathtt{OB}_{(u_i\&u_i)|(u_j\&u_i)|(u_i\&u_k)|(u_j\&u_k)}\alpha \leftrightarrow$$

$$\mathtt{OB}_{u_i|(u_j\&u_k)|(u_i\&u_j)|(u_i\&u_k)}\alpha \leftrightarrow \mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{(u_j\&u_k)}\alpha \wedge \mathtt{OB}_{(u_i\&u_j)}\alpha \wedge \mathtt{OB}_{(u_i\&u_k)}\alpha$$

Thus, we have

$\vdash \mathtt{OB}_{(u_i|u_j)\&(u_i|u_k)}\alpha \leftrightarrow \mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{(u_j\&u_k)} \wedge \mathtt{OB}_{(u_i\&u_j)}\alpha \wedge \mathtt{OB}_{(u_i\&u_k)}\alpha$ (I)

Considering this theorem and axiom $\vdash A\wedge B \rightarrow A$, by applying MP, we get $\vdash \mathtt{OB}_{(u_i|u_j)\&(u_i|u_k)}\alpha \rightarrow \mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{(u_j\&u_k)}\alpha$, and thus, $\mathtt{OB}_{(u_i|u_j)\&(u_i|u_k)}\alpha \rightarrow \mathtt{OB}_{u_i|(u_j\&u_k)}\alpha$.

[Only If Part] By axiom CAD, we can prove

$\vdash \mathtt{OB}_{u_i}\alpha \rightarrow \mathtt{OB}_{(u_i\&u_j)}\alpha$ (II)

$\vdash \mathtt{OB}_{u_i}\alpha \rightarrow \mathtt{OB}_{(u_i\&u_k)}\alpha$ (III)

Also by axiom JAD, we get

$\vdash \mathtt{OB}_{u_i|(u_j\&u_k)}\alpha \rightarrow \mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{(u_j\&u_k)}\alpha$ (IV)

By tautology $\vdash A \wedge B \rightarrow A$ and theorems (II), (III), (IV) in the above, we get

$\vdash \mathtt{OB}_{u_i|(u_j\&u_k)}\alpha \rightarrow \mathtt{OB}_{(u_i\&u_j)}\alpha$ (V)

$\vdash \mathtt{OB}_{u_i|(u_j\&u_k)}\alpha \rightarrow \mathtt{OB}_{(u_i\&u_k)}\alpha$ (VI)

Considering the inference rule $\frac{A\rightarrow B, A\rightarrow C}{A\rightarrow B\wedge C}$ (which is easily provable by tautologies and rule MP), and theorems (IV), (V), and (VI), we conclude that

$\vdash \mathtt{OB}_{u_i|(u_j\&u_k)}\alpha \rightarrow \mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{(u_j\&u_k)}\alpha \wedge \mathtt{OB}_{(u_i\&u_j)}\alpha \wedge \mathtt{OB}_{(u_i\&u_k)}\alpha$

By theorem (I), it is clear that $\vdash \mathtt{OB}_{(u_i|u_j)\&(u_i|u_k)}\alpha \rightarrow \mathtt{OB}_{u_i|(u_j\&u_k)}\alpha$. ■

Note that the collaboration (&) and disjunction (|) operators are not distributive over the delegation operator ($\rhd$); however, we have the following theorem, which is easily proven using DAD and JAD axioms.

- $\vdash \mathtt{OB}_{u_i|(u_j\rhd u_k)}\alpha \leftarrow \mathtt{OB}_{(u_i|u_j)\rhd(u_i|u_k)}\alpha$

4.3.4 Hierarchy of Authorities

Studying the behavior of authorities shows that authorities can form a hierarchy based on their strictness level. There exists a strictness relation between two authorities u_i and u_j, denoted by $u_i \leq u_j$ if and only if an authority u_j is stricter than u_i. Intuitively, this means that every case that u_i enacts as obligatory, u_j enacts as well. Furthermore, the equivalence relation is defined between two authorities (denoted by $u_i \approx u_j$) when they have the same strictness level.

Formally, we define the strictness relation and equivalence relation as follows.

- $\vdash u_i \leq u_j$ if and only if $\vdash (\mathtt{OB}_{u_i}\alpha \rightarrow \mathtt{OB}_{u_j}\alpha)$ (STD)
- $\vdash u_i \approx u_j$ if and only if $\vdash (\mathtt{OB}_{u_i}\alpha \leftrightarrow \mathtt{OB}_{u_j}\alpha)$ (EQD)

In other words, $u_i \approx u_j$ if and only if $u_i \leq u_j$ and $u_j \leq u_i$.

Using the Kripke structure, more strictness of u_j in comparison with u_i means that the set of norms of u_j is a subset of norms of u_i. On this basis, the interpretation of the strictness relation $u_i \leq u_j$ is defined as follows.

$$\Phi(u_i \leq u_j) = \begin{cases} W & \text{, if } \Lambda(u_j) \subseteq \Lambda(u_i) \\ \varnothing & \text{, otherwise} \end{cases}$$

Following the above definition of the strictness relation, the following theorems hold.

- $\vdash (u_i|u_j) \leq u_i$ and $\vdash (u_i|u_j) \leq u_j$
- $\vdash u_i \leq (u_i \& u_j)$ and $\vdash u_j \leq (u_i \& u_j)$

These relations mean that the authority obtained by collaborative composition of u_i and u_j (i.e., $u_i \& u_j$) is stricter than each of authorities u_i and u_j. However, both u_i and u_j are stricter than their disjunctive composition (i.e., $u_i|u_j$).

The above mentioned discussion concludes that the collaborative composition of two authorities u_i and u_j (i.e., $u_i \& u_j$) and their disjunctive composition (i.e., $u_i|u_j$) are the least upper bound (or supremum) and the greatest lower bound (or infimum) of u_i and u_j with respect to the strictness relation ($\leq$). On this basis, it is clear that $(\langle \mathcal{U}^*, \leq \rangle, \&, |)$ or the set of authorities with strictness relation and two operations & and $|$, forms a *lattice*.

In this lattice, if we have a strictness relation between two authorities (e.g., $u_i \leq u_j$), the following relations are easily concluded.

$$\text{if } \vdash u_i \leq u_j \text{ then } \begin{cases} u_i \& u_j \approx u_j \\ u_i | u_j \approx u_i \end{cases}$$

4.3.5 Conditions on Worlds Relations

In order to validate axiom DAI (i.e., $\texttt{OB}_{(u_i \rhd u_i)}\alpha \rightarrow \texttt{OB}_{u_i}\alpha$), relations $\Lambda(u_i)$ are required to be *transitive*. Transitivity means:

$$\forall w, w', w", \langle w, w' \rangle \in \Lambda(u_i) \wedge \langle w', w" \rangle \in \Lambda(u_i) \rightarrow \langle w, w" \rangle \in \Lambda(u_i)$$

The problem exists for transitivity is that, if we suppose that Λ for primitive authorities is transitive, the composition of authorities does not preserve the transitivity. Table 4.3.5 shows where we lose this property in the composition of authorities.

Table 4.2: Preserving the properties of worlds reachability relations in the composition of authorities.

Property	$\Lambda(u_1 \| u_2)$	$\Lambda(u_1 \& u_2)$	$\Lambda(u_1 \rhd u_2)$
Transitivity	$\times$	✓	$\times$
Seriality	✓	$\times$	✓
Reflexivity	✓	✓	✓

To solve the problem, considering the fact that idempotency of delegation operator for composite authorities is not required in practice, we eliminate the DAI axiom from the proposed proof theory and instead add the equivalences of the form $(u_i \rhd u_i) \approx u_i$ for all primitive authorities u_i (i.e., $u_i \in \mathcal{U}$) to the knowledge base of the model.

Another property that relations $\Lambda(u_i)$ require to have is *seriality* to validate the $\texttt{OB}$-MD axiom. As is shown in Table 4.3.5, seriality is not preserved by collaborative composition of authorities. For solving the problem, we require to have a stricter condition to be preserved in all cases. For this purpose, we take the *reflexivity* condition for the relations $\Lambda(u_i)$. It is easy to prove that this condition is preserved in all kinds of authority composition (see Table 4.3.5).

Although reflexivity tackles the problem of validating axiom $\texttt{OB}$-MD, it creates a new problem that the obtained semantics makes the proof theory incomplete. By the obtained semantics, the formula $\texttt{OB}_{u_i}\alpha \rightarrow \alpha$ for arbitrary authority u_i and arbitrary formula α (known as axiom T in modal logics) is satisfiable; however, it is not derivable by the proposed proof theory. In fact, we prefer not to have such an axiom in our logic, because this axiom bans authorities of a shared domain to have contradictory policy rules for an access. For example, if we have $\texttt{OB}_{u_1}do(s,o,a)$ and also $\texttt{IM}_{u_2}do(s,o,a)$, by this axiom we get $do(s,o,a)$ and $\neg do(s,o,a)$, which are contradictory. Thus, the set $\{\texttt{OB}_{u_1}do(s,o,a), \texttt{IM}_{u_2}do(s,o,a)\}$ becomes inconsistent, which is not desired in our approach. Thus, as is mentioned in completeness proof for MA(DL)$^2[^{\mathcal{U}}_{-}]$, we should remove collaborative composition operator (&) from this logic to solve this problem and have a complete logic.

4.4 MA(DL)$^2[^{-}_{\mathcal{D}}]$ Logic

In the overall framework proposed for authorization in SAEs in Section 3.4, the environment is divided into the set of security domains, which are not considered in the MA(DL)^2 language. To enrich Core MA(DL)^2 for specification and inference of security policies in distributed manner in SAEs, we add the concept of *security domain* (call it in short *domain*) to this logic. Note that in Core MA(DL)^2, the security policy rules specified by an authority for different domains cannot be distinguished from each other, and an authority cannot have different and contrary policies for different domains; however, in MA(DL)$^2[^{-}_{\mathcal{D}}]$ we can specify different policies for different security domains.

Security Domain is an abstract concept, which can be defined based on different factors, such as geographical situation, organizational ownership of resources, network security zoning, or coverage limitations in cellular networks and services. Each authority can specify security policies at each security domain, theoretically. However, in practice we have some limitations on this issue.

4.4.1 Syntax

To add security domains to Core MA(DL)^2, we add an enumerable set $\mathcal{D}$ of (security) domains to the alphabet of the core logic, and hence, the signature $\mathcal{S}$ is changed to a 7-tuple $\mathcal{S} = \langle \Sigma, \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{I}, \mathcal{U}, \mathcal{D} \rangle$. By having domains, the modal formula in the definition of c-formula is changed to $\texttt{ds}_{u@d}\alpha$ that means an authority u enacts the status $\texttt{ds}$ for α at domain d.

Since in practice, there exist inclusion relation between security domains, the notation $\preceq$ is used for this purpose. $d' \preceq d$ denotes that domain d' is a subdomain of d. The inclusion relation ($\preceq$) forms a hierarchy over the domains appear in $\mathcal{D}$.

Following the mentioned changes in the syntax of MA(DL)$^2[^{-}_{\mathcal{D}}]$ in comparison with Core MA(DL)^2, security policy rules are redefined in this logic as follows.

Definition 4.4.1 (Security Policy Rule at a Domain) *A security policy rule specified by an authority u at a security domain d is defined as $\alpha \rightarrow \texttt{ds}_{u@d}\beta$.*

4.4.2 Semantics

To present the semantics of MA(DL)$^2[^{-}_{\mathcal{D}}]$, we extend the Kripke model of MA(DL)^2 to 7-tuple $\mathcal{M} = \langle W, \Lambda, \Delta, I, \Phi, \mathbb{D}, J \rangle$, where

- W, Δ, and I are remained unchanged.
- $\mathbb{D}$ is a finite set of domain elements.
- $J = \mathcal{D} \rightarrow 2^{\mathbb{D}}$ is an interpretation function that assigns to each domain a set of domain elements.
- Λ is the enhanced of previous one that maps each authority to a function from domain elements to binary relations on the possible worlds. In other words, Λ for primitive authorities is defined as $\lambda = \mathcal{U} \rightarrow \{f : \mathbb{D} \rightarrow 2^{(W \times W)}\}$.
- Φ is extended and changed as follows

$$\Phi(\texttt{OB}_{u@d}\alpha) = \{w | \forall x \in J(d), \forall w' \in W, \langle w, w' \rangle \in \Lambda(u)(x) \rightarrow w' \in \Phi(\alpha)\}$$

$$\Phi(d' \preceq d) = \begin{cases} W & \text{, if } J(d') \subseteq J(d); \\ \varnothing & \text{, otherwise} \end{cases}$$

For clarifying the semantics represented for domains, see Figure 4.5. This figure illustrates the function $\Lambda(u)$ for the authority u. Some of the statements of u over the two domains d

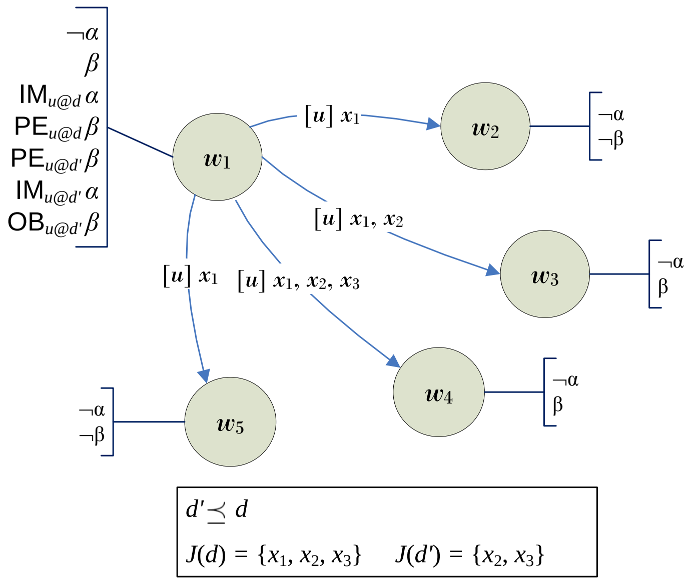


Figure 4.5: An example of the semantics of domains and statements over them.

and d' that are satisfiable in world w_1 are shown. As an example, you can see that $\mathtt{PE}_{u@d}\beta$ is satisfiable at world w_1, because for each domain element x_1, x_2, and x_3 of domain d, there exist at least one world accessible from w_1 where β is true. It is clear that similar statement (i.e., $\mathtt{PE}_{u@d'}\beta$) is satisfiable for subdomain d'. This example is an intuitive proof for corollary 4.4.1 in the proof theory of the logic of domains.

4.4.3 Proof Theory

The proof theory of MA(DL)$^2[^{-}_{\mathcal{D}}]$ is obtained by substituting the $\mathtt{OB}_u\alpha$ with to $\mathtt{OB}_{u@d}\alpha$ in the proof theory of Core MA(DL)^2. Furthermore, the following axioms should be added to it for taking the relations of domains (and their subdomains) into account.

- $\vdash d' \preceq d \rightarrow \wedge\mathtt{OB}_{u@d}\alpha \rightarrow \mathtt{OB}_{u@d'}\alpha$ (SDP)
- $\vdash d \preceq d$ (SDI)
- $\vdash d' \preceq d \wedge d'' \preceq d' \rightarrow d'' \preceq d$ (SDT)
- $\vdash d' \preceq d \rightarrow \mathtt{OB}_{u@d''}(d' \preceq d)$ (SDO)

Axiom SDP says that any obligation holds for a domain, holds for its subdomains as well. Axioms SDI and SDT show the reflexivity and transitivity of domain inclusion relation. Axiom SDO says a subdomain relationship is necessarily acceptable by all authorities in all domains.

By the above axiomatic system, the following corollary is concluded.

Corollary 4.4.1 *Suppose there exist security domains d, d', and d'' such that $d' \preceq d \preceq d''$ and we have $\mathtt{ds}_{u@d}\alpha$ for domain d. If $\mathtt{ds} \in \{\mathtt{OB}, \mathtt{IM}\}$, then the statement holds for subdomain d' as well (i.e., $\vdash \mathtt{ds}_{u@d'}\alpha$ for $\mathtt{ds} \in \{\mathtt{OB}, \mathtt{IM}\}$), and if $\mathtt{ds} \in \{\mathtt{PE}, \mathtt{GR}\}$, then the statement holds for superdomain d'' as well (i.e., $\vdash \mathtt{ds}_{u@d''}\alpha$ for $\mathtt{ds} \in \{\mathtt{PE}, \mathtt{GR}\}$).*

Proof. It is easy to prove. ∎

4.5 MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ Logic

Core MA(DL)^2 augmenting with logic of authorities and logic of security domains becomes MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$. In other words, MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ is the result of the combination of MA(DL)$^2[^{\mathcal{U}}_{-}]$ and MA(DL)$^2[^{-}_{\mathcal{D}}]$. Thus, in this logic we can have norms in the form of $\mathtt{ds}_{u@d}\alpha$, where u is a primitive or composite authority, d is a security domain, and $\mathtt{ds} \in \{\mathtt{OB}, \mathtt{IM}, \mathtt{PE}, \mathtt{GR}\}$ is a deontic status.

The semantics of MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ is resulted from the combination of the semantics of MA(DL)$^2[^{\mathcal{U}}_{-}]$ presented in Section 4.3.2 and semantics of MA(DL)$^2[^{-}_{\mathcal{D}}]$ presented in Section 4.4.2. Thus, a 7-tuple Kripke style model $\mathcal{M} = \langle W, \Lambda, \Delta, I, \Phi, \mathbb{D}, J\rangle$ is taken into account.The definition of all elements of $\mathcal{M}$ except Λ is easy and clear. Thus, we just redefine Λ here.

$$\begin{aligned}
&\lambda = \mathcal{U} \to \{f : \mathbb{D} \to 2^{(W\times W)}\} \qquad\qquad \Lambda(u) = \lambda(u), \text{if } u \in \mathcal{U}\\
&\Lambda(u_i|u_j)(x) = \Lambda(u_i)(x) \cup \Lambda(u_j)(x)\\
&\Lambda(u_i \& u_j)(x) = \Lambda(u_i)(x) \cap \Lambda(u_j)(x)\\
&\Lambda(u_i \rhd u_j)(x) = \Lambda(u_i)(x) \circ \Lambda(u_j)(x)
\end{aligned}$$

Due to the change in the definition of Λ, the strictness relation on authorities is changed as follows.

$$\Phi(u_i \leq u_j) = \begin{cases} W & \text{, if } \forall x \in \mathbb{D}, \Lambda(u_j)(x) \subseteq \Lambda(u_i)(x) \\ \varnothing & \text{, otherwise} \end{cases}$$

By the above interpretation, if $u_i \leq u_j$, then authority u_j in all security domains is stricter than authority u_i. We can interpret the strictness relation in other way, in which for each security domain, strictness relation is defined independent from other domains. For this purpose, we use notation $\leq_d$ for a security domain, and interpret it as follows.

$$\Phi(u_i \leq_d u_j) = \begin{cases} W & \text{, if } \forall x \in J(d), \Lambda(u_j)(x) \subseteq \Lambda(u_i)(x) \\ \varnothing & \text{, otherwise} \end{cases}$$

The proof theory of MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ logic is obtained from the combination of the axioms and inference rules presented for MA(DL)$^2[^{\mathcal{U}}_{-}]$ and MA(DL)$^2[^{-}_{\mathcal{D}}]$ in sections 4.3.3 and 4.4.3. Note that we need to substitute $\mathtt{OB}_u p$ with $\mathtt{OB}_{u@d} p$ in the axioms we inherit from the axiomatic system of MA(DL)$^2[^{\mathcal{U}}_{-}]$.

4.6 Summary

For security policy specification and inference in SAEs, we need to have an appropriate logical language. In this chapter, we introduced the MA(DL)^2 logic, which is our proposed logic for this purpose. MA(DL)^2 is a logic family, whose core is resulted from the integrating description logic with n-ary predicates on concepts, and multi-authority (poly-modal) version of standard deontic logic. Extending Core MA(DL)^2 by logic of composite authorities (employed for cooperative security administration in shared subdomains) results in MA(DL)$^2[^{\mathcal{U}}_{-}]$, and by logic of security domains results in MA(DL)$^2[^{-}_{\mathcal{D}}]$ in this family. MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ in this logic family has all the capabilities of the other members.

Each member of MA(DL)^2 logic in introduced in this chapter by its syntax, semantics, and proof theory. The properties of these logics, including soundness, completeness, decidability, expressive power, and time complexity of satisfiability problem are investigated in the next section.

Chapter 5

Properties of MA(DL)^2 Logic Family

By introducing each logic, it is required to investigate the properties such as soundness, completeness, decidability, expressive power, and computational complexity. After introducing the MA(DL)^2 logic family in previous chapter, we get through the properties of this logic family in this chapter.

Note that in this chapter, to simplify the proofs, we eliminated concept types from MA(DL)^2. Such an assumption does not spoil the presented proofs, since for each set of typed MA(DL)^2 formulae, we can generate a set of untyped MA(DL)^2 formulae. To this aim, for example, for each type σ, we take a concept $\top_\sigma$, and for each concept $C : \sigma$ in the formulae, we add $C \sqsubseteq \top_\sigma$. Also, we replace each complemented concept $\neg C$ with $\top_\sigma \sqcap \neg C$. By generating such an equivalent set of formulae in untyped MA(DL)^2, we can easily prove that for each model $\mathcal{M}$ satisfying untyped formulae, we can construct a model $\mathcal{M}'$, which satisfied the typed formulae, and vice versa.

5.1 Soundness

To prove that an inference system works correctly, we should prove that the system is sound. An axiom is *sound* if and only if it is valid. An inference rule is sound if and only if it preserves the soundness, i.e. it maps valid premises to valid conclusions. In an axiomatic logical system, the whole system is sound if and only if its all axioms and inference rules are sound. In other words, a logic is sound if $\vdash \alpha$ implies that $\vDash \alpha$; if a formula α is provable, then it is valid.

5.1.1 Soundness of Core MA(DL)^2

The soundness proofs of axioms OB-MK, OB-MD, SPP, and UPP in the Core MA(DL)^2 logic are presented in the following lemmas. The soundness proofs of the other axioms and inference rules are no more complicated.

Lemma 5.1.1 (Soundness of Axiom OB-MK) *The OB-MK axiom is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\vDash_w^{\mathcal{M}} \mathtt{OB}_u(p \to q)$, which holds iff by clause (iv) of proposition 4.2.1, in all $w' \in \Lambda(u)(w), \vDash_{w'}^{\mathcal{M}} p \to q$ iff in all $w' \in \Lambda(u)(w)$, if $\vDash_{w'}^{\mathcal{M}} p$, then $\vDash_{w'}^{\mathcal{M}} q$, which is equal to this sentence that if in all $w' \in \Lambda(u)(w), \vDash_{w'}^{\mathcal{M}} p$, then in all $w' \in \Lambda(u)(w), \vDash_{w'}^{\mathcal{M}} q$ as well. By clause (iv) of proposition 4.2.1, this implies that if $\vDash_w^{\mathcal{M}} \mathtt{OB}_u p$, then $\vDash_w^{\mathcal{M}} \mathtt{OB}_u q$, which holds iff $\vDash_w^{\mathcal{M}} \mathtt{OB}_u p \to \mathtt{OB}_u q$.

Thus, by assumption $\vDash_w^{\mathcal{M}} \mathtt{OB}_u(p \to q)$, we conclude that $\vDash_w^{\mathcal{M}} \mathtt{OB}_u p \to \mathtt{OB}_u q$. Hence, by clause (i) of proposition 4.2.1, for every model $\mathcal{M}$ and every world $w \in W$ in $\mathcal{M}$, we have $\vDash_w^{\mathcal{M}} \mathtt{OB}_u(p \to q) \to (\mathtt{OB}_u p \to \mathtt{OB}_u q)$, which means axiom OB-MK is valid and sound. ■

Lemma 5.1.2 (Soundness of Axiom OB-MD) *The OB-MD axiom is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\vDash_w^{\mathcal{M}} \mathtt{OB}_u p$, which holds iff by clause (iv) of proposition 4.2.1, in all $w' \in \Lambda(u)(w), \vDash_{w'}^{\mathcal{M}} p$, or in other words in all $w' \in \Lambda(u)(w), \nvDash_{w'}^{\mathcal{M}} \neg p$, which implies that $\nvDash_w^{\mathcal{M}} \mathtt{OB}_u \neg p$ and also $\vDash_w^{\mathcal{M}} \neg\mathtt{OB}_u \neg p$.

Hence, by assumption $\vDash_w^{\mathcal{M}} \mathtt{OB}_u p$, we conclude that $\vDash_w^{\mathcal{M}} \neg\mathtt{OB}_u \neg p$, which results in $\vDash_w^{\mathcal{M}} \mathtt{OB}_u p \to \neg\mathtt{OB}_u \neg p$. Thus, axiom OB-MD is valid and sound. ■

Lemma 5.1.3 (Soundness of Axiom SPP) *The SPP axiom is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\vDash_w^{\mathcal{M}} C_i' \sqsubseteq C_i$, which holds iff by the definition of truth, $w \in \Phi(C_i' \sqsubseteq C_i)$ iff by the definition of Φ in semantics of Core MA(DL)^2, $[\![C_i']\!]_w^I \subseteq [\![C_i]\!]_w^I$.

Also, suppose $\vDash_w^{\mathcal{M}} p(C_1, ..., C_i, ..., C_n)$. Thus, $w \in \Phi(p(C_1, ..., C_i, ..., C_n))$, and by the definition of Φ, $\prod_{j=1}^{n} [\![C_j]\!]_w^I \subseteq [\![p]\!]_w^I$. In combination with the first assumption (i.e., $[\![C_i']\!]_w^I \subseteq [\![C_i]\!]_w^I$), we get the following relation:

$$\prod_{j=1}^{i-1} [\![C_j]\!]_w^I \times [\![C_i']\!]_w^I \times \prod_{j=i+1}^{n} [\![C_j]\!]_w^I \subseteq \prod_{j=1}^{n} [\![C_j]\!]_w^I \subseteq [\![p]\!]_w^I$$

Since $\prod_{j=1}^{i-1} [\![C_j]\!]^I_w \times [\![C_i{}']\!]^I_w \times \prod_{j=i+1}^{n} [\![C_j]\!]^I_w \subseteq [\![p]\!]^I_w$, by the definition of Φ, we get $w \in \Phi(p(C_1, ..., C_i{}', ..., C_n))$, and hence, $\vDash^{\mathcal{M}}_w p(C_1, ..., C_i{}', ..., C_n)$.

Therefore, if $\vDash^{\mathcal{M}}_w C_i{}' \sqsubseteq C_i$, then if $\vDash^{\mathcal{M}}_w p(C_1, ..., C_i, ..., C_n)$, then $\vDash^{\mathcal{M}}_w p(C_1, ..., C_i{}', ..., C_n)$. By clause (i) of proposition 4.2.1, the above sentence means $\vDash^{\mathcal{M}}_w C_i{}' \sqsubseteq C_i \rightarrow$ $(p(C_1, ..., C_i, ..., C_n) \rightarrow p(C_1, ..., C_i{}', ..., C_n))$. Thus, axiom SPP is sound. ■

Lemma 5.1.4 (Soundness of Axiom UPP) *The UPP axiom is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\vDash^{\mathcal{M}}_w p(C_1, ..., C_i, ..., C_n) \wedge$ $p(C_1, ..., C_i{}', ..., C_n)$, which holds iff by the definition of truth we have $w \in \Phi(p(C_1, ..., C_i, ..., C_n))$ and $w \in \Phi(p(C_1, ..., C_i{}', ..., C_n))$. This holds iff by the definition of Φ in semantics of Core MA(DL)^2 we have

$$\prod_{j=1}^{n} [\![C_j]\!]^I_w \subseteq [\![p]\!]^I_w$$

$$\prod_{j=1}^{i-1} [\![C_j]\!]^I_w \times [\![C_i{}']\!]^I_w \times \prod_{j=i+1}^{n} [\![C_j]\!]^I_w \subseteq [\![p]\!]^I_w$$

Since, if $A \subseteq C$ and $B \subseteq C$, then $A \cup B \subseteq C$, based on the above equation, we get $\left(\prod_{j=1}^{i-1} [\![C_j]\!]^I_w \times [\![C_i]\!]^I_w \times \prod_{j=i+1}^{n} [\![C_j]\!]^I_w \cup \prod_{j=1}^{i-1} [\![C_j]\!]^I_w \times [\![C_i{}']\!]^I_w \times \prod_{j=i+1}^{n} [\![C_j]\!]^I_w\right) \subseteq [\![p]\!]^I_w$.

Since $A \times (B \cup C) = (A \times B) \cup (A \times C)$, we conclude $\prod_{j=1}^{i-1} [\![C_j]\!]^I_w \times ([\![C_i]\!]^I_w \cup [\![C_i{}']\!]^I_w) \times$ $\prod_{j=i+1}^{n} [\![C_j]\!]^I_w \subseteq [\![p]\!]^I_w$, and thus, $\prod_{j=1}^{i-1} [\![C_j]\!]^I_w \times [\![C_i \sqcup C_i{}']\!]^I_w \times \prod_{j=i+1}^{n} [\![C_j]\!]^I_w \subseteq [\![p]\!]^{\mathcal{M}}_w$

Now, by the definition of Φ, we have $w \in \Phi(p(C_1, ..., C_i \sqcup C_i{}', ..., C_n))$, and hence $\vDash^{\mathcal{M}}_w$ $p(C_1, ..., C_i \sqcup C_i{}', ..., C_n)$. By clause (i) of proposition 4.2.1, the above sentence concludes

$$\vDash^{\mathcal{M}}_w p(C_1, ..., C_i, ..., C_n) \wedge p(C_1, ..., C_i{}', ..., C_n) \rightarrow p(C_1, ..., C_i \sqcup C_i{}', ..., C_n)$$

Hence, axiom UPP is valid and sound. ■

Theorem 5.1.1 (Soundness of Core MA(DL)^2) *The Core MA(DL)^2 logic is sound; if* $\vdash \alpha$, *then* $\vDash \alpha$.

Proof. Lemmas 5.1.1, 5.1.2, 5.1.3, and 5.1.4 prove the soundness of axioms OB-MK, OB-MD, SPP, and UPP as examples. The soundness proof of other axioms and inference rules are no more complicated. Thus, by induction on the length of proof, we can easily prove that the Core MA(DL)^2 logic is sound. ■

5.1.2 Soundness of MA(DL)$^2[^{\mathcal{U}}_{-}]$

The soundness proofs of axioms JAD, DAD, CAD, and DDC in the MA(DL)$^2[^{\mathcal{U}}_{-}]$ logic are presented in the following lemmas. The soundness proofs of other axioms are easy.

Lemma 5.1.5 (Soundness of Axiom JAD) *Axiom JAD in MA(DL)$^2[^{\mathcal{U}}_{-}]$ is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\models^{\mathcal{M}}_{w} \mathtt{OB}_{(u_i|u_j)}\alpha$. Thus, $w \in \Phi(\mathtt{OB}_{(u_i|u_j)}\alpha)$ iff $w \in \Phi(\mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{u_j}\alpha)$, because

$$\begin{aligned}\Phi(\mathtt{OB}_{(u_i|u_j)}\alpha) &= \{w|\Lambda(u_i|u_j)(w) \subseteq \Phi(\alpha)\} = \{w|(\Lambda(u_i) \cup \Lambda(u_j))(w) \subseteq \Phi(\alpha)\}\\ &= \{w|\Lambda(u_i)(w) \cup \Lambda(u_j)(w) \subseteq \Phi(\alpha)\} = \{w|\Lambda(u_i)(w) \subseteq \Phi(\alpha) \wedge \Lambda(u_j)(w) \subseteq \Phi(\alpha)\}\\ &= \{w|\Lambda(u_i)(w) \subseteq \Phi(\alpha)\} \cap \{w|\Lambda(u_j)(w) \subseteq \Phi(\alpha)\} = \Phi(\mathtt{OB}_{u_i}\alpha) \cap \Phi(\mathtt{OB}_{u_j}\alpha)\\ &= \Phi(\mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{u_j}\alpha)\end{aligned}$$

Hence, $\models^{\mathcal{M}}_{w} \mathtt{OB}_{(u_i|u_j)}\alpha \leftrightarrow \mathtt{OB}_{u_i}\alpha \wedge \mathtt{OB}_{u_j}\alpha$. ■

Lemma 5.1.6 (Soundness of Axiom DAD) *Axiom DAD in MA(DL)$^2[^{\mathcal{U}}_{-}]$ is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\models^{\mathcal{M}}_{w} \mathtt{OB}_{(u_i \rhd u_j)}\alpha$. Thus, $w \in \Phi(\mathtt{OB}_{(u_i \rhd u_j)}\alpha)$ if and only if $w \in \Phi(\mathtt{OB}_{u_i}(\mathtt{OB}_{u_j}\alpha))$, because:

$$\begin{aligned}\Phi(\mathtt{OB}_{(u_i \rhd u_j)}\alpha) &= \{w|\Lambda(u_i \rhd u_j)(w) \subseteq \Phi(\alpha)\} = \{w|(\Lambda(u_i) \circ \Lambda(u_j))(w) \subseteq \Phi(\alpha)\}\\ &= \{w|\forall w', \text{if } \langle w, w' \rangle \in \Lambda(u_i) \text{ then, } \Lambda(u_j)(w') \subseteq \Phi(\alpha)\}\\ &= \{w|\Lambda(u_i)(w) \subseteq \{w'|\Lambda(u_j)(w') \subseteq \Phi(\alpha)\}\}\\ &= \{w|\Lambda(u_i)(w) \subseteq \Phi(\mathtt{OB}_{u_j}\alpha)\} = \Phi(\mathtt{OB}_{u_i}(\mathtt{OB}_{u_j}\alpha))\end{aligned}$$

Hence, $\models^{\mathcal{M}}_{w} \mathtt{OB}_{(u_i \rhd u_j)}\alpha \leftrightarrow \mathtt{OB}_{u_i}(\mathtt{OB}_{u_j}\alpha)$. ■

Lemma 5.1.7 (Soundness of Axiom CAD) *Axiom CAD in MA(DL)$^2[^{\mathcal{U}}_{-}]$ is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\models_w^{\mathcal{M}} \texttt{OB}_{u_i}\alpha \vee \texttt{OB}_{u_j}\alpha$. Then:

$$\begin{aligned}
&w \in \Phi(\texttt{OB}_{u_i}\alpha \vee \texttt{OB}_{u_j}\alpha) \textbf{ iff } w \in \Phi(\texttt{OB}_{u_i}\alpha) \cup \Phi(\texttt{OB}_{u_j}\alpha)\\
&\textbf{iff } w \in \{w|\forall w', \langle w, w'\rangle \in \Lambda(u_i) \to w' \in \Phi(\alpha)\} \cup \{w|\forall w', \langle w, a'\rangle \in \Lambda(u_j) \to w' \in \Phi(\alpha)\}\\
&\textbf{then } w \in \{w|\forall w', \langle w, w'\rangle \in \Lambda(u_i) \cap \Lambda(u_j) \to w' \in \Phi(\alpha)\} \cup\\
&\qquad \{w|\forall w', \langle w, a'\rangle \in \Lambda(u_j) \cap \Lambda(u_i) \to w' \in \Phi(\alpha)\}\\
&\textbf{iff } w \in \{w|\langle w, w'\rangle \in \Lambda(u_i \& u_j) \to w' \in \Phi(\alpha)\} \quad \textbf{iff } w \in \Phi(\texttt{OB}_{(u_i \& u_j)}\alpha)
\end{aligned}$$

Thus, $\models_w^{\mathcal{M}} \texttt{OB}_{(u_i \& u_j)}\alpha$. Hence, the axiom is sound. ■

Lemma 5.1.8 (Soundness of Axiom DDC) *Axiom DDC in MA(DL)$^2[\frac{\mathcal{U}}{-}]$ is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\models_w^{\mathcal{M}} \texttt{OB}_{(u_i \rhd (u_j \& u_k))}\alpha$. Then:

$$\begin{aligned}
&w \in \Phi(\texttt{OB}_{(u_i \rhd (u_j \& u_k))})\textbf{iff } \forall w', [\langle w, w'\rangle \in \Lambda(u_i) \circ \Lambda(u_j \& u_k)] \to w' \in \Phi(\alpha)\\
&\textbf{iff } \forall w', [\langle w, w'\rangle \in \Lambda(u_i) \circ (\Lambda(u_j) \cap \Lambda(u_k))] \to w' \in \Phi(\alpha)\\
&\textbf{iff } \forall w', [\exists w", \langle w, w"\rangle \in \Lambda(u_i) \wedge (\langle w", w'\rangle \in \Lambda(u_j) \wedge \langle w", w'\rangle \in \Lambda(u_k))] \to w' \in \Phi(\alpha)
\end{aligned}$$

Considering the fact that $\exists x.(p(x) \wedge q(x)) \to \exists x.p(x) \wedge \exists x.q(x)$, by the above, we get

$$\begin{aligned}
&\forall w', [(\exists w", \langle w, w"\rangle \in \Lambda(u_i) \wedge \langle w", w'\rangle \in \Lambda(u_j)) \wedge\\
&\qquad (\exists w", \langle w, w"\rangle \in \Lambda(u_i) \wedge \langle w", w'\rangle \in \Lambda(u_k))] \to w' \in \Phi(\alpha)\\
&\textbf{iff } \forall w', [\langle w, w'\rangle \in \Lambda(u_i) \circ \Lambda(u_j) \wedge \langle w, w'\rangle \in \Lambda(u_i) \circ \Lambda(u_k)] \to w' \in \Phi(\alpha)\\
&\textbf{iff } \forall w', [\langle w, w'\rangle \in \Lambda(u_i \rhd u_j) \wedge \langle w, w'\rangle \in \Lambda(u_i \rhd u_k)] \to w' \in \Phi(\alpha)\\
&\textbf{iff } \forall w', \langle w, w'\rangle \in \Lambda(u_i \rhd u_j) \cap \Lambda(u_i \rhd u_k) \to w' \in \Phi(\alpha)\\
&\textbf{iff } \forall w', \langle w, w'\rangle \in \Lambda((u_i \rhd u_j) \& (u_i \rhd u_k)) \to w' \in \Phi(\alpha)\\
&\textbf{iff } w \in \Phi(\texttt{OB}_{(u_i \rhd u_j) \& (u_j \rhd u_k)}\alpha)
\end{aligned}$$

Thus, we have $\models_w^{\mathcal{M}} \texttt{OB}_{(u_i \rhd u_j) \& (u_j \rhd u_k)}\alpha$. By clause (i) of Proposition 4.2.1, we conclude that axiom DDC is valid and sound. ■

Theorem 5.1.2 (Soundness of MA(DL)$^2[^{\mathcal{U}}_{-}]$) *MA(DL)$^2[^{\mathcal{U}}_{-}]$ logic is sound such that $\vdash \alpha$ then $\vDash \alpha$.*

Proof. Lemmas 5.1.5, 5.1.6, 5.1.7, and 5.1.8 prove the soundness of axioms JAD, DAD, CAD, and DDC respectively. The soundness proofs of other axioms are easy. Thus, by induction on the length of proof, we can easily prove that the MA(DL)$^2[^{\mathcal{U}}_{-}]$ logic is sound. ■

5.1.3 Soundness of MA(DL)$^2[^{-}_{\mathcal{D}}]$

The most important axiom in MA(DL)$^2[^{-}_{\mathcal{D}}]$ is the SDP axiom. In this section, we fist prove the soundness of axiom SDP, and then present the theorem of MA(DL)$^2[^{-}_{\mathcal{D}}]$ soundness.

Lemma 5.1.9 (Soundness of Axiom SDP) *Axiom SDP in MA(DL)$^2[^{-}_{\mathcal{D}}]$ is sound.*

Proof. Suppose for some model $\mathcal{M}$ and some world $w \in W$ in $\mathcal{M}$, we have $\vDash^{\mathcal{M}}_{w} d' \preceq d$ and $\vDash^{\mathcal{M}}_{w} \mathtt{OB}_{u@d}\alpha$. Thus, $w \in \mathtt{OB}_{u@d}\alpha$ and by the definition of Φ we have

$$\forall x \in J(d), \forall w' \in W, \langle w, w' \rangle \in \Lambda(u)(x) \rightarrow w' \in \Phi(\alpha)$$

Since $\vDash^{\mathcal{M}}_{w} d' \preceq d$, by the semantics we have $J(d') \subseteq J(d)$ and hence

$$forallx \in J(d'), \forall w' \in W, \langle w, w' \rangle \in \Lambda(u)(x) \rightarrow w' \in \Phi(\alpha)$$

By the presented semantics, the above equation concludes that $w \in \Phi(\mathtt{OB}_{u@d'}\alpha)$; in other words $\vDash^{\mathcal{M}}_{w} \mathtt{OB}_{u@d'}\alpha$.

Hence, by suppose that $\vDash^{\mathcal{M}}_{w} d \preceq d' \wedge \mathtt{OB}_{u@d}\alpha$, we conclude that $\vDash^{\mathcal{M}}_{w} \mathtt{OB}_{u@d'}\alpha$. By clause (i) of Proposition 4.2.1, we get that axiom SDP is valid and sound. ■

Theorem 5.1.3 (Soundness of MA(DL)$^2[^{-}_{\mathcal{D}}]$) *The MA(DL)$^2[^{-}_{\mathcal{D}}]$ logic is sound such that $\vdash \alpha$ then $\vDash \alpha$.*

Proof. The soundness of axiom SDP is presented in Lemma 5.1.9. The soundness proofs of the other axioms are similar. Thus, by induction on the length of proof, we can easily prove that the MA(DL)$^2[^{-}_{\mathcal{D}}]$ logic is sound. ■

5.1.4 Soundness of MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$

In previous sections, the soundness of the fragments of MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ are proved. The soundness of the axioms of MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ can be presented analogously. In fact, we just need to change some notations in the provided proofs in previous sections. Thus, we have the following theorem as the result.

Theorem 5.1.4 (Soundness of MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$) *The MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ logic is sound, i.e., if* $\vdash \alpha$*, then* $\vDash \alpha$.

Proof. By the above description, it is evident. ■

5.2 Completeness

Completeness is equivalent to the satisfiability of every consistent set of formulae. The weaker version of completeness shows that the proof theory of a logic is enough to infer all the valid formulae, i.e. if $\vDash \alpha$ then $\vdash \alpha$.

To prove the completeness of the MA(DL)^2 logic, we follow the approach proposed by Henkin [79] and its adaptation [11] that is used to prove the completeness of deontic logic. In this approach, we take an arbitrary consistent set Γ of formulae and construct a model that satisfies Γ. In this model, the possible worlds are maximal consistent sets of formulae and there exists a world that contains Γ and also satisfies Γ.

5.2.1 Completeness of Core MA(DL)^2

We begin the proof by some definitions and lemmas. Note that, although some definitions and lemmas related to the maximal consistent sets might be found in many references, for having a complete and detailed proof, we present all of them here.

Definition 5.2.1 (Maximal Consistent Set) *A set* Ψ *of formulae is a maximal consistent set iff*

(i) Ψ *is consistent,*

(ii) if $\Psi \subseteq \Theta$ *and* Θ *is consistent, then* $\Psi = \Theta$.

Lemma 5.2.1 (Existence of Maximal Consistent Set) *For each consistent set Γ of formulae, there exists a maximal consistent set, denoted by Γ^*, with $\Gamma \subseteq \Gamma^*$.*

Proof. Suppose that there is a list of all formulae in MA(DL)^2 including $\alpha_1, \alpha_2, \cdots$. We define a sequence of sets of formulae as follows:

- $\Gamma_0 = \Gamma$
- $\Gamma_{n+1} = \begin{cases} \Gamma_n \cup \{\alpha_n\} & \text{, if } \Gamma_n \cup \{\alpha_n\} \text{ is consistent} \\ \Gamma_n & \text{, otherwise} \end{cases}$

We define $\Gamma^* = \bigcup_{n \geq 0} \Gamma_n$. It is obvious that Γ^* is consistent and $\Gamma \subseteq \Gamma^*$.

Now, we show that Γ^* is a maximal consistent set, too.
Let Ψ be a consistent set of formulae and $\Gamma^* \subseteq \Psi$. For every $x \in \Psi$, there exists α_n in the list of MA(DL)^2 formulae such that $\alpha_n = x$. Since $\Gamma_n \subseteq \Gamma^* \subseteq \Psi$ and Ψ is consistent, $\Gamma_n \cup \{\alpha_n\}$ is also consistent. Thus, $\Gamma_{n+1} = \Gamma_n \cup \{\alpha_n\}$ that implies $\alpha_n \in \Gamma_{n+1} \subseteq \Gamma^*$. Since $\alpha_n = x$, we conclude that for every $x \in \Psi$, we have $x \in \Gamma^*$ (i.e., $\Psi \subseteq \Gamma^*$). Therefore, $\Gamma^* = \Psi$, and hence Γ^* is a maximal consistent set. ■

Lemma 5.2.2 (Properties of Maximal Consistent Sets) *Let Ψ be a maximal consistent set. Then, we have:*

(i) for each formula α, either $\alpha \in \Psi$ or $\neg\alpha \in \Psi$

(ii) Ψ is closed under inference, i.e., if $\Psi \vdash \alpha$, then $\alpha \in \Psi$

(iii) $\neg\alpha \in \Psi$ iff $\alpha \notin \Psi$

(iv) $\alpha \wedge \beta \in \Psi$ iff $\alpha \in \Psi$ and $\beta \in \Psi$

(v) $\alpha \rightarrow \beta \in \Psi$ iff if $\alpha \in \Psi$, then $\beta \in \Psi$

Proof. To prove (i), we suppose that $\alpha \notin \Psi$ and $\neg\alpha \notin \Psi$. Since Ψ is consistent, there exists a consistent set $\Theta = \Psi \cup \{\alpha\}$ and thus $\Psi \subseteq \Theta$. By the definition of maximal consistent sets, we conclude that $\Psi = \Theta$. By the above definition of Θ, we have $\alpha \in \Theta$, and hence $\alpha \in \Psi$, which is a contradiction.

To prove (ii), we suppose that Ψ is not closed under inference that means there exists a formula α such that $\Psi \vdash \alpha$, but $\alpha \notin \Psi$. Thus, by clause (i) $\neg\alpha \in \Psi$ and hence $\Psi \vdash \neg\alpha$. Since $\Psi \vdash \alpha$ and $\Psi \vdash \neg\alpha$, we get $\Psi \vdash \mathtt{F}$. This shows Ψ is an inconsistent set, which is a contradiction.

The proofs of (iii) to (v) can be shown easily. ■

Definition 5.2.2 ($\Psi\downarrow^{u}_{\alpha}$ Notation) *Let u be an authority, α be a formula, and Ψ be a maximal consistent set such that $\neg \mathit{OB}_u\alpha \in \Psi$. We define a set $\Psi\downarrow^{u}_{\alpha}$ of formulae as follows:*

$$\Psi\downarrow^{u}_{\alpha} = \{\beta | \mathit{OB}_u\beta \in \Psi\} \cup \{\neg\alpha\}$$

Lemma 5.2.3 (Modified Makinson's Lemma) *Let u be an authority, α be a formula, and Ψ be a maximal consistent set such that $\neg \mathit{OB}_u\alpha \in \Psi$. Then, $\Psi\downarrow^{u}_{\alpha}$ is a consistent set.*

Proof. Suppose $\Psi\downarrow^{u}_{\alpha}$ is not a consistent set. Then, there are formulae like $\beta_1, \beta_2, ..., \beta_n$ $(n \geq 0)$ such that $\mathtt{OB}_u\beta_i \in \Psi$ and by the definition of $\Psi\downarrow^{u}_{\alpha}$, we get $\vdash (\beta_1 \wedge \beta_2 \wedge ... \wedge \beta_n \wedge \neg\alpha) \rightarrow \mathtt{F}$. Hence, by axiom schema TAUT, we have $\vdash (\beta_1 \wedge \beta_2 \wedge ... \wedge \beta_n) \rightarrow \alpha$.

For case n=0, this results in $\vdash \alpha$ and by the OB-MO axiom schema $\vdash \mathtt{OB}_u\alpha$. Hence, by clause (ii) of lemma 5.2.2, $\mathtt{OB}_u\alpha \in \Psi$. Also $\neg\mathtt{OB}_u\alpha \in \Psi$ by the hypothesis. Thus, both $\mathtt{OB}_u\alpha$ and $\neg\mathtt{OB}_u\alpha$ are in Ψ, which is a contradiction.

For case $n \geq 1$, $\vdash (\beta_1 \wedge \beta_2 \wedge ... \wedge \beta_n) \rightarrow \alpha$ results in $\vdash \beta_1 \rightarrow (\beta_2 \rightarrow ...(\beta_n \rightarrow \alpha)...)$. Hence, by OB-MO, we have $\vdash \mathtt{OB}_u(\beta_1 \rightarrow (\beta_2 \rightarrow ...(\beta_n \rightarrow \alpha)...))$ and by the axiom OB-MK, $\vdash \mathtt{OB}_u\beta_1 \rightarrow \mathtt{OB}_u(\beta_2 \rightarrow ...(\beta_n \rightarrow \alpha)...)$. By repeating the above steps $n-1$ times, we obtain $\vdash \mathtt{OB}_u\beta_1 \rightarrow (\mathtt{OB}_u\beta_2 \rightarrow ...(\mathtt{OB}_u\beta_n \rightarrow \mathtt{OB}_u\alpha)...)$. Hence, by applying n times clause (v) of lemma 5.2.2, and considering the fact that for all $1 \leq i \leq n$, $\mathtt{OB}_u\beta_i \in \Psi$, we get $\mathtt{OB}_u\alpha \in \Psi$. Again, both $\mathtt{OB}_u\alpha$ and $\neg\mathtt{OB}_u\alpha$ are in Ψ, which is a contradiction. ■

The partitions resulted by $\equiv$ relationships defined in Γ (set of the MA(DL)^2 formulae) is denoted by $\hat{\mathcal{C}}_\Gamma$, which is defined as follows.

$$\hat{\mathcal{C}}_\Gamma = \{[C]_\Gamma |\ C \in \mathcal{C}\}$$
$$[C]_\Gamma = \{C' |\ C \equiv C' \in \Gamma^* \wedge \quad C, C' \in \mathcal{C}\}$$

Note that in the above definition of each partition $[C]_\Gamma$, we used Γ^* instead of Γ. It is due to that fact that Γ^* is closed under inference and includes all the equivalent concepts of each concept C.

By the definition of $\hat{\mathcal{C}}_\Gamma$, all the preliminaries for presenting the canonical model are provided. Note that $\hat{\mathcal{C}}_\Gamma$ is used in the definition of $\Delta_\mathcal{C}$ in the canonical model.

Definition 5.2.3 (Canonical Model) *Let Γ be a consistent set. By lemma 5.2.1 there exists a maximal consistent set Γ^* such that $\Gamma \subseteq \Gamma^*$. The* canonical model generated by Γ *is a Kripke structure $\mathcal{CM} = \langle W_\mathcal{C}, \Lambda_\mathcal{C}, \Delta_\mathcal{C}, I_\mathcal{C}, \Phi_\mathcal{C}\rangle$ where:*

(i) $W_\mathcal{C}$ is defined as the smallest collection of maximal consistent sets such that

- *Γ^* is in $W_\mathcal{C}$,*
- *if w is in $W_\mathcal{C}$ and α is a formula with $\neg \mathtt{OB}_u\alpha \in w$, then $(w{\downarrow}^u_\alpha)^*$ is in $W_\mathcal{C}$.*

Note that each world $w \in W_\mathcal{C}$ is a maximal consistent set obtained through the above definition.

(ii) $\Lambda_\mathcal{C}$ is the mapping function from authorities to a set of binary relations on $W_\mathcal{C}$ such that for all authorities u and all $w_i, w_j \in W_\mathcal{C}$, $\langle w_i, w_j\rangle \in \Lambda_\mathcal{C}(u)$ iff for all formulae α, whenever $\mathtt{OB}_u\alpha \in w_i$, then $\alpha \in w_j$.

(iii) $\Delta_\mathcal{C}$ is a non-empty set of objects in which for each concept $C_i \in \hat{\mathcal{C}}_\Gamma$, there exists a unique object a_i, and for each instance d_j appears in Γ, there exists a unique object a_j. Since we have unique name assumption (UNA) in this logic, two instances cannot refer to the same object, and thus we should take for each instance a unique object in $\Delta_\mathcal{C}$. Note that $\Delta_\mathcal{C}$ at least contains the aforementioned members (and thus it is a non-empty set) and it may contain other members, too.

(iv) $I_\mathcal{C}$ is an interpretation function such that at each world $w \in W_\mathcal{C}$ is defined as follows. Considering the consistency of the formulae existing in Γ and completeness of the proof theory derived from [136] for the inference of c-formulae and a-formulae, we can claim that for c-formulae and a-formulae existing in Γ, there is a model (interpretation) J in $\mathcal{ALC}$ for concepts, roles, and instances such that

- $\Gamma \vdash C_i \sqsubseteq C_j$ *iff* $[\![C_i]\!]^J \subseteq [\![C_j]\!]^J$
- $\Gamma \vdash C_i \equiv C_j$ *iff* $[\![C_i]\!]^J = [\![C_j]\!]^J$

- $\Gamma \vdash C(a)$ *iff* $[\![a]\!]^J \in [\![C]\!]^J$
- $\Gamma \vdash R(a,b)$ *iff* $\langle [\![a]\!]^J, [\![b]\!]^J \rangle \in [\![R]\!]^J$

Now, we define the interpretation function $I_{\mathcal{C}}$ in each world $w \in W_{\mathcal{C}}$ by extending the interpretation function J as follows.

- *For each atomic concept C_i, and each role R_i, and each instance a_i in Γ, $I_{\mathcal{C}}$ is defined as $[\![C_i]\!]_w^{I_{\mathcal{C}}} = [\![C_i]\!]^J$, $[\![R_i]\!]_w^{I_{\mathcal{C}}} = [\![R_i]\!]^J$, and $[\![a_i]\!]_w^{I_{\mathcal{C}}} = [\![a_i]\!]^J$, respectively.*
- *For non-atomic complex concepts, $I_{\mathcal{C}}$ is defined inductively as mentioned in semantics of Core MA(DL)^2 in Section 4.2.2.*
- *For each n-ary predicate symbol (name) p in the language of Γ, $I_{\mathcal{C}}$ is defined as $[\![p]\!]_w^{I_{\mathcal{C}}} = \bigcup\limits_{p(C_1,\ldots,C_n) \in w} \prod\limits_{i=1}^{n} [\![C_i]\!]_w^{I_{\mathcal{C}}}$.*

(v) $\Phi_{\mathcal{C}}$ *is a mapping function that maps each formula to a subset of $W_{\mathcal{C}}$, and is defined as follows.*

- *For propositions, it is defined by $\phi_{\mathcal{C}}$; for each proposition x (appeared in the language of Γ) and each $w \in W_{\mathcal{C}}$, $w \in \phi_{\mathcal{C}}(x)$ iff $x \in w$.*
- *For* **T** *and* **F**, *it is defined as $\Phi_{\mathcal{C}}(\mathtt{T}) = W_{\mathcal{C}}$ and $\Phi_{\mathcal{C}}(\mathtt{F}) = \varnothing$.*
- *For n-ary predicates on concepts $C_1, \ldots, C_n$, such as $p(C_1, \ldots, C_n)$, it is defined as follows:*

$$\Phi_{\mathcal{C}}(p(C_1, \ldots, C_n)) = \{w | \prod_{i=1}^{n} [\![C_i]\!]_w^{I_{\mathcal{C}}} \subseteq [\![p]\!]_w^{I_{\mathcal{C}}}\}$$

- *For c-formulae, it is defined as follows:*

$$\Phi_{\mathcal{C}}(C \sqsubseteq C') = \begin{cases} W & \text{,if for all } w \in W_{\mathcal{C}}, [\![C]\!]_w^{I_{\mathcal{C}}} \subseteq [\![C']\!]_w^{I_{\mathcal{C}}} \\ \varnothing & \text{,otherwise} \end{cases}$$

$$\Phi_{\mathcal{C}}(C \equiv C') = \begin{cases} W & \text{,if for all } w \in W_{\mathcal{C}}, [\![C]\!]_w^{I_{\mathcal{C}}} = [\![C']\!]_w^{I_{\mathcal{C}}} \\ \varnothing & \text{,otherwise} \end{cases}$$

- *for a-formulae, it is defined as follows:*

$$\Phi_{\mathcal{C}}(C(a)) = \begin{cases} W & \text{,if for all } w \in W_{\mathcal{C}}, [\![a]\!]_w^{I_\mathcal{C}} \in [\![C]\!]_w^{I_\mathcal{C}} \\ \varnothing & \text{,otherwise} \end{cases}$$

$$\Phi_{\mathcal{C}}(R(a,b)) = \begin{cases} W & \text{,if for all } w \in W_{\mathcal{C}}, \langle [\![a]\!]_w^{I_\mathcal{C}}, [\![b]\!]_w^{I_\mathcal{C}} \rangle \in [\![R]\!]_w^{I_\mathcal{C}} \\ \varnothing & \text{,otherwise} \end{cases}$$

- *For non-atomic formulae, it is defined inductively similar to function Φ in Section 4.2.2.*

In the following, after proving some required properties of the canonical models for completeness proof, we verify that the defined canonical model is a valid Kripke model of the Core MA(DL)^2 logic.

Lemma 5.2.4 *Let Γ be a consistent set and $\mathcal{CM} = \langle W_\mathcal{C}, \Lambda_\mathcal{C}, \Delta_\mathcal{C}, I_\mathcal{C}, \Phi_\mathcal{C}\rangle$ be a canonical model generated by Γ. For all $w \in W_\mathcal{C}$ and all formulae α, we have*

(i) $\mathtt{OB}_u\alpha \in w$ iff for all $w' \in W_\mathcal{C}$, if $\langle w, w'\rangle \in \Lambda_\mathcal{C}(u)$, then $\alpha \in w'$,

(ii) $\mathtt{PE}_u\alpha \in w$ iff there exists $w' \in W_\mathcal{C}$ such that $\langle w, w'\rangle \in \Lambda_\mathcal{C}(u)$ and $\alpha \in w'$.

Proof. They are proved as follows.

(i) [Only If Part] By the definition of $\Lambda_\mathcal{C}$, $\mathtt{OB}_u\alpha \in w$ implies that for all $w' \in W_\mathcal{C}$, if $\langle w, w'\rangle \in \Lambda_\mathcal{C}(u)$, then we have $\alpha \in w'$.

[If Part] We suppose that $\mathtt{OB}_u\alpha \notin w$. Thus, $\neg\mathtt{OB}_u\alpha \in w$, and by the definition of $w{\downarrow}_\alpha^u$, $\neg\alpha \in w{\downarrow}_\alpha^u$, and since $w{\downarrow}_\alpha^u \subseteq (w{\downarrow}_\alpha^u)^*$, thus $\neg\alpha \in (w{\downarrow}_\alpha^u)^*$ and $\alpha \notin (w{\downarrow}_\alpha^u)^*$. Also, by the definition of $W_\mathcal{C}$, $(w{\downarrow}_\alpha^u)^* \in W_\mathcal{C}$ and by the definition of $w{\downarrow}_\alpha^u$, for all formulae β, if $\mathtt{OB}_u\beta \in w$, then $\beta \in (w{\downarrow}_\alpha^u)^*$. Thus, by the definition of $\Lambda_\mathcal{C}$, $\langle w, (w{\downarrow}_\alpha^u)^*\rangle \in \Lambda_\mathcal{C}(u)$. Hence, there exists a world $w' = (w{\downarrow}_\alpha^u)^* \in W_\mathcal{C}$ such that $\langle w, w'\rangle \in \Lambda_\mathcal{C}(u)$ and $\alpha \notin w'$, which contradicts the hypothesis.

(ii) [Only If Part] If $\mathtt{PE}_u\alpha \in w$, then by axiom PE-Def and inference rule MP, $\neg\mathtt{OB}_u\neg\alpha \in w$. By the definition of $w{\downarrow}_\alpha^u$, we have $\alpha \in (w{\downarrow}_\alpha^u)^*$, and thus, by the definition of $W_\mathcal{C}$,

$(w{\downarrow}^u_\alpha)^* \in W_\mathcal{C}$. Hence, similar to the previous part, we can conclude that there exists a world $w' = (w{\downarrow}^u_\alpha)^* \in W_\mathcal{C}$, such that $\langle w, w' \rangle \in \Lambda_\mathcal{C}(u)$ and $\alpha \in w'$.

[If Part] Suppose $\mathtt{PE}_u\alpha \notin w$, then $\neg\mathtt{PE}_u\alpha \in w$, and thus, $\mathtt{OB}_u\neg\alpha \in w$. By clause (i), this implies that for all $w' \in W_\mathcal{C}$, if $\langle w, w' \rangle \in \Lambda_\mathcal{C}(u)$, then $\neg\alpha \in w'$, i.e., $\alpha \notin w'$, which contradicts the hypothesis.

■

Lemma 5.2.5 *Let Γ be a consistent set and $\mathcal{CM} = \langle W_\mathcal{C}, \Lambda_\mathcal{C}, \Delta_\mathcal{C}, I_\mathcal{C}, \Phi_\mathcal{C} \rangle$ be a canonical model generated by Γ. For each c-formula and a-formula α, we have*

(i) if $\Gamma \vdash \alpha$, then for each $w \in W_\mathcal{C}$, we have $\alpha \in w$, and

(ii) if there exists a world $w \in W_\mathcal{C}$ such that $\alpha \in w$, then $\Gamma \vdash \alpha$.

Proof.

(i) If $\alpha = C \sqsubseteq C'$, by induction on the definition of possible worlds in $W_\mathcal{C}$, we prove the above lemma.

Basis: if $\Gamma \vdash C \sqsubseteq C'$, since maximal consistent set Γ^* is closed under inference, we have $C \sqsubseteq C' \in \Gamma^*$.

Induction Step: if we suppose that $C \sqsubseteq C' \in w$ and $\neg\mathtt{OB}_u\beta \in w$, then there exists $w' = (w{\downarrow}^u_\beta)^*$ such that $w' \in W_\mathcal{C}$. Now, we prove that $C \sqsubseteq C' \in w'$.

Since $C \sqsubseteq C' \in w$, by axiom SOB and inference rule MP, formula $\mathtt{OB}_u(C \sqsubseteq C')$ is inferable. Since w as a maximal consistent set is closed under inference, thus $\mathtt{OB}_u(C \sqsubseteq C') \in w$. Now, if $\neg\mathtt{OB}_u\beta \in w$, then by the definition of $W_\mathcal{C}$, we have $(w{\downarrow}^u_\beta)^* \in W_\mathcal{C}$. By the definition of $w{\downarrow}^u_\beta$, because $\mathtt{OB}_u(C \sqsubseteq C') \in w$, we have $C \sqsubseteq C' \in w{\downarrow}^u_\beta$ and thus, $C \sqsubseteq C' \in (w{\downarrow}^u_\beta)^*$.

Hence, by induction, we can prove that for each world w appearing in constructing set $W_\mathcal{C}$, if $\Gamma \vdash C \sqsubseteq C'$, then $C \sqsubseteq C' \in w$.

The proof of clause (i), for $\alpha = C \equiv C'$ is obvious, considering the proof presented for $\alpha = C \sqsubseteq C'$. For $\alpha = C(a)$ and $\alpha = R(a, b)$ we have similar proofs. Note that in these two cases, axioms COB and ROB are used instead of SOB.

(ii) If $\alpha = C \sqsubseteq C'$ and there exists $w' = (w{\downarrow}^u_\beta)^* \in W_\mathcal{C}$ such that $C \sqsubseteq C' \in w'$, then by the definition $w{\downarrow}^u_\beta$, we have $w \vdash \mathtt{OB}_u(C \sqsubseteq C')$ and thus, by inference rule OB-MO and considering the fact that w (as a maximal consistent set) is closed under inference, we have $C \sqsubseteq C' \in w$. Analogously, if we step backward the definition of worlds belonging to $W_\mathcal{C}$, we conclude that $C \sqsubseteq C' \in \Gamma^*$ and this means $\Gamma \vdash C \sqsubseteq C'$.

For case $\alpha = C \equiv C'$, $\alpha = C(a)$, and $\alpha = R(a, b)$ there exist similar proofs.

■

Lemma 5.2.6 *Let Γ be a consistent set and $\mathcal{CM} = \langle W_\mathcal{C}, \Lambda_\mathcal{C}, \Delta_\mathcal{C}, I_\mathcal{C}, \Phi_\mathcal{C}\rangle$ be a canonical model generated by Γ, then for each $w \in W_\mathcal{C}$ we have*

(i) for each $C_i, C_j \in \mathcal{C}^*$, $[\![C_i]\!]^{I_\mathcal{C}}_w \subseteq [\![C_j]\!]^{I_\mathcal{C}}_w$ iff $C_i \sqsubseteq C_j \in w$, and analogously $[\![C_i]\!]^{I_\mathcal{C}}_w = [\![C_j]\!]^{I_\mathcal{C}}_w$ iff $C_i \equiv C_j \in w$,

(ii) for each $C \in \mathcal{C}$ and $R \in \mathcal{R}$, $[\![a]\!]^{I_\mathcal{C}}_w \in [\![C]\!]^{I_\mathcal{C}}_w$ iff $C(a) \in w$, and analogously $\langle [\![a]\!]^{I_\mathcal{C}}_w, [\![b]\!]^{I_\mathcal{C}}_w\rangle \in [\![R]\!]^{I_\mathcal{C}}_w$ iff $R(a, b) \in w$,

(iii) for each n-ary predicate symbol p and concepts $C_i \in \mathcal{C}^*(1 \leq i \leq n)$, $\prod_{i=1}^{n} [\![C_i]\!]^{I_\mathcal{C}}_w \subseteq [\![p]\!]^{I_\mathcal{C}}_w$ iff $p(C_1, ..., C_n) \in w$.

Proof.

(i, ii) By the definition of $I_\mathcal{C}$, $[\![C_i]\!]^{I_\mathcal{C}}_w \subseteq [\![C_j]\!]^{I_\mathcal{C}}_w$ iff $\Gamma \vdash C_i \sqsubseteq C_j$ and by Lemma 5.2.5 we have $\Gamma \vdash C_i \sqsubseteq C_j$ iff $C_i \sqsubseteq C_j \in w$. For other c-formulae and a-formulae, i.e., $C_i \equiv C_j$, $C(a)$, $R(a, b)$, we have similar proofs.

(iii) [If Part] We prove that, if $p(C_1, ..., C_n) \in w$, then $\prod_{i=1}^{n} [\![C_i]\!]^{I_\mathcal{C}}_w \subseteq [\![p]\!]^{I_\mathcal{C}}_w$.

By the definition of $I_\mathcal{C}$ for n-ary predicates, we have $[\![p]\!]^{I_\mathcal{C}}_w = \bigcup_{p(C_1,...,C_n)\in w} \prod_{i=1}^{n} [\![C_i]\!]^{I_\mathcal{C}}_w$, and thus, for every $p(C_1, ..., C_n) \in w$, we have $\prod_{i=1}^{n} [\![C_i]\!]^{I_\mathcal{C}}_w \subseteq [\![p]\!]^{I_\mathcal{C}}_w$.

[Only If Part] We prove that, if $\prod_{i=1}^{n} [\![C_i]\!]^{I_\mathcal{C}}_w \subseteq [\![p]\!]^{I_\mathcal{C}}_w$, then $p(C_1, ..., C_n) \in w$.

If $\prod_{i=1}^{n} [\![C_i]\!]^{I_\mathcal{C}}_w = \varnothing$, implies that one of the arguments of the n-ary predicate is $\bot$. Thus, by axiom BP and by the fact that w as a maximal consistent set is closed under inference, we have $p(...,\bot,...) \in w$ and the lemma holds. If $\prod_{i=1}^{n} [\![C_i]\!]_w^{I_C} \neq \varnothing$, for all C_i in $p(C_1,...,C_n)$, we have $C_i \neq \bot$. Now we prove the lemma in two cases $n = 1$ and $n > 1$. For case $n = 1$, we prove that if $[\![C]\!]_w^{I_C} \subseteq [\![p]\!]_w^{I_C}$, then $p(C) \in w$. By holding $[\![C]\!]_w^{I_C} \subseteq [\![p]\!]_w^{I_C}$, the following cases are possible:

1. There exists $C_i \in \mathcal{C}^*$ such that $[\![C]\!]_w^{I_C} \subseteq [\![C_i]\!]_w^{I_C}$ and $p(C_i) \in w$. By the clause (i) of this lemma, since $[\![C]\!]_w^{I_C} \subseteq [\![C_i]\!]_w^{I_C}$, we conclude that $C \sqsubseteq C_i \in w$ and by the fact that $p(C_i) \in w$, axiom SPP, and inference rule MP, we can infer that $p(C) \in w$.
2. There exist $C_1,...,C_k \in \mathcal{C}^*(k > 1)$ such that $[\![C]\!]_w^{I_C} \subseteq \bigcup_{i=1}^{k} [\![C_i]\!]_w^{I_C}$ and for every $1 \leq i \leq k$, we have $p(C_i) \in w$. In this case, since $[\![C]\!]_w^{I_C} \subseteq \bigcup_{i=1}^{k} [\![C_i]\!]_w^{I_C}$, by the clause (i) of this lemma, we have $C \sqsubseteq \bigsqcup_{i=1}^{k} C_i$. By the fact that for every $1 \leq i \leq k$, we have $p(C_i) \in w$, by axiom UPP and inference rule MP, we infer that $p(\bigsqcup_{i=1}^{k} C_i) \in w$. Hence, since $C \sqsubseteq \bigsqcup_{i=1}^{k} C_i$ by the previous case we infer that $p(C) \in w$.

For case $n > 1$, we prove that if $\prod_{i=1}^{n} [\![C_i]\!]_w^{I_C} \subseteq [\![p]\!]_w^{I_C}$, then $p(C_1,...,C_n) \in w$. By holding $\prod_{i=1}^{n} [\![C_i]\!]_w^{I_C} \subseteq [\![p]\!]_w^{I_C}$, the following cases are possible:

1. There exist $C_{j1},...,C_{jn} \in \mathcal{C}^*$ such that for every $1 \leq i \leq n$ we have $[\![C_{ji}]\!]_w^{I_C} \subseteq [\![C_i]\!]_w^{I_C}$ and $p(C_{j1},...,C_{jn}) \in w$. Now by axiom SPP and inference rule MP (for every argument of p), we conclude that $p(C_1,...,C_n) \in w$.
2. For each argument i of p, there exist $C_{i1},...,C_{ik} \in \mathcal{C}^*$ such that $[\![C_i]\!]_w^{I_C} \subseteq \bigcup_{j=1}^{k} [\![C_{ij}]\!]_w^{I_C}$. Similar to case (2) of $n = 1$, we conclude that $p(C_1,...,C_n) \in w$.

■

Lemma 5.2.7 (Verification of the Canonical Model) *Canonical model $\mathcal{CM}$ generated by Γ set of Core MA(DL)^2 formulae, introduced in Definition 5.2.3, is a Kripke model of MA(DL)^2.*

Proof. We prove that the definition of the elements $W_{\mathcal{C}}, \Lambda_{\mathcal{C}}, \Delta_{\mathcal{C}}, I_{\mathcal{C}}$, and $\Phi_{\mathcal{C}}$ of the canonical model match the definition of the elements W, Λ, Δ, I, and Φ of the Kripke model of MA(DL)^2 introduced in Section 4.2.2.

(i) By taking each maximal consistent set as a possible world, it is clear that $W_{\mathcal{C}}$ is a set of possible worlds, and it is non-empty, because at least Γ^* is in $W_{\mathcal{C}}$.

(ii) By the definition of canonical model, $\Lambda_{\mathcal{C}}$ is a mapping function that maps each authority to a binary relation on $W_{\mathcal{C}}$. Now, we just need to prove that all relations $\Lambda_{\mathcal{C}}(u)$ are serial.

By taking one of the axioms (introduced in proof theory of MA(DL)^2) as α, and by inference rule OB-MO, for an arbitrary authority u, we get for all $w \in W_{\mathcal{C}}$, $\mathtt{OB}_u\alpha \in w$ (because w as a maximal consistent set is closed under inference). Hence, by axiom OB-MD and inference rule MP, we have $\neg\mathtt{OB}_u\neg\alpha \in w$, or $\mathtt{PE}_u\alpha \in w$. Then, by Lemma 5.2.4, we conclude that there exists $w' \in W_{\mathcal{C}}$, such that $\langle w, w' \rangle \in \Lambda_{\mathcal{C}}(u)$ and $\alpha \in w'$. Therefore, for all $w \in W_{\mathcal{C}}$, there exists $w' \in W_{\mathcal{C}}$ such that $\langle w, w' \rangle \in \Lambda_{\mathcal{C}}(u)$, i.e., $\Lambda_{\mathcal{C}}(u)$ is serial.

(iii) It is obvious that the definition of $\Delta_{\mathcal{C}}$ matches the definition of Δ.

(iv) It is clear that $I_{\mathcal{C}}$ satisfies the condition mentioned for I, and the based on the definition of $I_{\mathcal{C}}$, interpretation of all c-formulae and a-formulae are the same in the all possible worlds.

(v) It is obvious that $\Phi_{\mathcal{C}}$ matches the definition of Φ.

■

Lemma 5.2.8 (Coincidence) *For each formula α, and for each maximal consistent set $w \in W_{\mathcal{C}}$ in a canonical model $\mathcal{CM}$, $\vDash_w^{\mathcal{CM}} \alpha$ iff $\alpha \in w$.*

Proof. We prove the lemma by induction on the length of α.
<u>Basis:</u> if α is an atomic formula

- $\alpha = \mathtt{T}$. Then, $\vDash_w^{\mathcal{CM}} \mathtt{T}$ iff due to the fact that w as a maximal consistent set is closed under inference, by axiom TAUT, we have $\mathtt{T} \in w$. Correspondingly, we have $\nvDash_w^{\mathcal{CM}} \mathtt{F}$ iff $\mathtt{F} \notin w$.

- $\alpha = x_i$ is a proposition. Then, $\models_w^{\mathcal{CM}} x_i$ iff by the definition of truth, $w \in \Phi_{\mathcal{C}}(x_i)$ iff by the definition of $\Phi_{\mathcal{C}}$ for propositions, $w \in \phi_{\mathcal{C}}(x_i)$, and by the definition of $\phi_{\mathcal{C}}$, $x_i \in w$.

- $\alpha = C_i \sqsubseteq C_j$ is a subsumption relationship. Then, $\models_w^{\mathcal{CM}} C_i \sqsubseteq C_j$ iff by the definition of truth $[\![C_i]\!]_w^{I_{\mathcal{C}}} \subseteq [\![C_j]\!]_w^{I_{\mathcal{C}}}$ iff by clause (i) of Lemma 5.2.6, we have $C_i \sqsubseteq C_j \in w$, i.e., $\alpha \in w$. If $\alpha = C_i \equiv C_j$, similarly we have $\models_w^{\mathcal{CM}} C_i \equiv C_j$ iff $C_i \equiv C_j \in w$.

- $\alpha = p(C_1, ..., C_n)$ is an n-ary predicate on atomic or complex concepts. Then, $\models_w^{\mathcal{CM}} p(C_1, ..., C_n)$ iff by the definition of truth, $w \in \Phi_{\mathcal{C}}(p(C_1, ..., C_n))$ iff by the definition of $\Phi_{\mathcal{C}}$, $\prod\limits_{i=1}^{n} [\![C_i]\!]_w^{I_{\mathcal{C}}} \subseteq [\![p]\!]_w^{I_{\mathcal{C}}}$, iff by clause (ii) of Lemma 5.2.6, $p(C_1, ..., C_n) \in w$, i.e., $\alpha \in w$.

- $\alpha = C(a)$ is an assertional formula. Then, by the definition of truth, $\models_w^{\mathcal{CM}} C(a)$ iff $[\![a]\!]_w^{I_{\mathcal{C}}} \in [\![C]\!]_w^{I_{\mathcal{C}}}$ iff by clause (iii) of Lemma 5.2.6 we have $C(a) \in w$. If $\alpha = R(a, b)$, similarly we have $\models_w^{\mathcal{CM}} R(a, b)$ iff $R(a, b) \in w$.

Induction Step: suppose that the lemma holds for β with length n, i.e., $\models_w^{\mathcal{CM}} \beta$ iff $\beta \in w$. Now, we prove the lemma for formulae with $n + 1$.

- By Proposition 4.2.1, and Lemma 5.2.2 on properties of maximal consistent sets, the proof of the lemma for inductive cases $\neg\alpha, \alpha_i \wedge \alpha_j$, and $\alpha_i \rightarrow \alpha_j$ can be shown easily.

- For case $\alpha = \texttt{OB}_u\beta$, we have $\models_w^{\mathcal{CM}} \texttt{OB}_u\beta$
  iff by Proposition 4.2.1, for all $w' \in \Lambda_{\mathcal{C}}(u)(w)$, $\models_{w'}^{\mathcal{CM}} \beta$
  iff by the induction hypothesis, for all $w' \in \Lambda_{\mathcal{C}}(u)(w)$, $\beta \in w'$
  iff by clause (i) of Lemma 5.2.4, we have $\texttt{OB}_u\beta \in w$.

- For cases $\alpha = \texttt{PE}_u\beta$, $\texttt{IM}_u\beta$, and $\texttt{GR}_u\beta$, we can easily prove by the result of the inductive cases $\texttt{OB}_u\beta$, $\beta_i \wedge \beta_j$, and $\neg\beta$.

■

Lemma 5.2.9 (Model Existence) *Let Γ be a set of formulae in Core MA(DL)^2. If Γ is consistent, then Γ is satisfiable.*

Proof. Assume $\mathcal{CM}$ be a canonical model generated by Γ. Hence, by the definition of $W_{\mathcal{C}}$ we have $\Gamma^* \in W_{\mathcal{C}}$. By the coincidence lemma, we have $\models_{\Gamma^*}^{\mathcal{CM}} \alpha$ iff $\alpha \in \Gamma^*$. Since $\Gamma \subseteq \Gamma^*$, we have $\models_{\Gamma^*}^{\mathcal{CM}} \alpha$ for every formula α in Γ. Thus, for consistent set Γ, we constructed a model, i.e.

$\mathcal{CM}$, such that for a world, i.e. Γ^*, we have $\models^{\mathcal{CM}}_{\Gamma^*} \alpha$ for every α in Γ. Hence, Γ is satisfiable. ■

Theorem 5.2.1 (Strong Completeness of Core MA(DL)^2) *Let Γ be a set of formulae in Core MA(DL)^2 and α be a formula in this logic. If $\Gamma \models \alpha$, then $\Gamma \vdash \alpha$.*

Proof. We first prove that the theorem holds iff the *model existence* lemma holds.

- (strong completeness theorem $\Rightarrow$ model existence lemma)
  Suppose Γ is a consistent set, thus, $\Gamma \nvdash \mathtt{F}$. By the strong completeness theorem we have $\Gamma \nvDash \mathtt{F}$. Hence, there exists a model $\mathcal{M}$, and a world w in this model such that $\models^{\mathcal{M}}_{w} \Gamma$, that means Γ is satisfiable. Therefore, the model existence lemma holds.

- (model existence lemma $\Rightarrow$ strong completeness theorem)
  Suppose Γ is a set of formulae and α is a formula in Core MA(DL)^2 such that $\Gamma \models \alpha$ and $\Gamma \nvdash \alpha$.
  $\Gamma \models \alpha$ holds iff for all worlds w, if $\models^{\mathcal{M}}_{w} \Gamma$, then $\models^{\mathcal{M}}_{w} \alpha$. By the definition of consistency, $\Gamma \nvdash \alpha$ means $\Gamma \cup \{\neg\alpha\}$ is consistent, and by the model existence lemma $\Gamma \cup \{\neg\alpha\}$ is satisfiable, that means there exists a model $\mathcal{M}$ and a world w such that $\models^{\mathcal{M}}_{w} \Gamma$ and $\models^{\mathcal{M}}_{w} \neg\alpha$. This contradicts the hypothesis.

In fact, the *strong completeness* theorem and the *model existence* lemma are equivalent. As is shown in lemma 5.2.9, the model existence lemma holds in Core MA(DL)^2, and thus, the strong completeness theorem holds too. ■

Corollary 5.2.1 (Weak Completeness of Core MA(DL)^2) *Let α be a formula in Core MA(DL)^2. If $\models \alpha$, then $\vdash \alpha$. In other words, all* valid *formulae are* provable.

Proof. Suppose $\Gamma = \varnothing$. By strong completeness, we have if $\varnothing \models \alpha$, then $\varnothing \vdash \alpha$. This immediately concludes that if $\models \alpha$, then $\vdash \alpha$. ■

5.2.2 Completeness of MA(DL)$^2[^{\mathcal{U}}_{-}]$ Logic

The discussion on the required conditions for the $\Lambda(u_i)$ relations in Section 4.3.5 shows that preserving some properties imposes some problems in completeness of the proposed proof theory. It is shown that the collaboration operator (&) does not preserve the seriality, and substituting the seriality with reflexivity (which implies seriality as well), makes the proposed logic incomplete.

Furthermore, the existence of the collaboration operator (interpreted as the intersection of worlds relations) results in some problems in having a complete proof theory regarding Kripke-style semantics. The similar problem in programs dynamic logic (PDL) with intersection was open for several years [16]. Note that there are many differences between dynamic logic and MA(DL)$^2[^{\mathcal{U}}_{-}]$ that deters using the proposed approach for having a complete proof theory for dynamic logic with intersection in the MA(DL)$^2[^{\mathcal{U}}_{-}]$ logic[1].

By the above discussion, we can conclude that we cannot have a complete proof theory that satisfies our logical requirements, except that we waive the collaboration composition from MA(DL)$^2[^{\mathcal{U}}_{-}]$. By such a modification in MA(DL)$^2[^{\mathcal{U}}_{-}]$ (call it MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$), we reach a complete logic regarding the presented semantics and proposed proof theory. Although, the expressive power of the logic is decreased in MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$, in addition to obtaining a complete proof theory, we can prove the decidability of MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$, and leverage the *analytical tableaux* method for automated reasoning in real applications.

To prove the completeness of MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$, we follow the approach presented in the previous section for Core MA(DL)^2. Thus, by preserving the definition of maximal consistent sets and their properties as well as the definition of $\Psi{\downarrow}^u_\alpha$, we define the canonical model in this logic as follows.

Definition 5.2.4 (Canonical Model in MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$) *Let Γ be a consistent set. By Lemma 5.2.1, a maximal consistent set Γ^* exists such that $\Gamma \subseteq \Gamma^*$. Thus, the canonical model generated by Γ is a Kripke model $\mathcal{CM} = \langle W_\mathcal{C}, \Lambda_\mathcal{C}, \Delta_\mathcal{C}, J_\mathcal{C}, I_\mathcal{C}, \Phi_\mathcal{C}\rangle$ where its all elements except $\Lambda_\mathcal{C}$ are as the same as defined in 5.2.3, and $\Lambda_\mathcal{C}$ is defined as:*

- *$\Lambda_\mathcal{C}$ is the mapping function from authorities to a set of binary relations on $W_\mathcal{C}$ such that for primitive authorities $u \in \mathcal{U}$ are defined by $\lambda_\mathcal{C}$. For all $w_i, w_j \in W_\mathcal{C}$, $\langle w_i, w_j\rangle \in \Lambda_\mathcal{C}(u)$ iff for all formulae α, whenever $\mathtt{OB}_u\alpha \in w_i$, then $\alpha \in w_j$. The definition of $\Lambda_\mathcal{C}$ is extended for composite authorities similar to the one defined in semantics of MA(DL)$^2[^{\mathcal{U}}_{-}]$ in Section 4.3.2.*

[1]Some of the main differences are requirement of having the OB-MD and DAI axioms in MA(DL)$^2[^{\mathcal{U}}_{-}]$, and philosophical differences between the notion of possible worlds and the accessibility relations in Kripke-style models of the two logics.

Regarding the above change in the definition of the canonical model, it is enough to refine the proof of Lemma 5.2.4 for MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$. The proof of clause (ii) of this lemma does not change, thus, we just prove the clause (i) of this lemma in the following.

Proof.[Clause (i) of Lemma 5.2.4]

[Only If Part] We prove it inductively as follows.

Basis ($u \in \mathcal{U}$): it is immediately obtained by the definition of $\Lambda_{\mathcal{C}}$ for primitive authorities.

Inductive Step ($u \in \mathcal{U}^*$): suppose the lemma holds for composite authorities u_1 and u_2.

- Case $u = (u_1|u_2)$: since w as a maximal consistent set is closed under inference, if $\mathtt{OB}_{(u_1|u_2)}\alpha \in w$, then $\mathtt{OB}_{u_1}\alpha, \mathtt{OB}_{u_2}\alpha \in w$. Due to the fact that the lemma holds for u_1 and u_2, and $\vdash ((p \to r) \wedge (q \to r) \equiv (p \vee q) \to r)$, we get $\forall w' \in W_{\mathcal{C}}, \langle w, w' \rangle \in \Lambda_{\mathcal{C}}(u_1) \vee (w, w') \in \Lambda_{\mathcal{C}}(u_2) \to \alpha \in w'$.

Since $\Lambda_{\mathcal{C}}(u_1|u_2) = \Lambda_{\mathcal{C}}(u_1) \cup \Lambda_{\mathcal{C}}(u_2)$, the lemma holds for authority $u = (u_1|u_2)$.

- Case $u = (u_1 \triangleright u_2)$: since w is closed under inference, if $\mathtt{OB}_{(u_1 \triangleright u_2)}\alpha \in w$, then by axiom DAD, we have $\mathtt{OB}_{u_1}(\mathtt{OB}_{u_2}\alpha) \in w$. By the induction hypothesis, $\forall w' \in W_{\mathcal{C}}, \langle w, w' \rangle \in \Lambda_{\mathcal{C}}(u_1) \to \mathtt{OB}_{u_2}\alpha \in w'$, and also because $\mathtt{OB}_{u_2}\alpha \in w'$, we have $\forall w'' \in W_{\mathcal{C}}, \langle w', w'' \rangle \in \Lambda_{\mathcal{C}}(u_2) \to \alpha \in w''$.
Thus, $\forall w' \in W_{\mathcal{C}}, \langle w, w' \rangle \in \Lambda_{\mathcal{C}}(u_1) \to (\forall w'' \in W_{\mathcal{C}}, (\langle w', w'' \rangle \in \Lambda_{\mathcal{C}}(u_2) \to \alpha \in w''))$
iff $\forall w'', \exists w', \langle w, w' \rangle \in \Lambda_{\mathcal{C}}(u_1) \to (\langle w', w'' \rangle \in \Lambda_{\mathcal{C}}(u_2) \to \alpha \in w'')$
iff $\forall w'', \exists w', \langle w, w' \rangle \in \Lambda_{\mathcal{C}}(u_1) \wedge \langle w', w'' \rangle \in \Lambda_{\mathcal{C}}(u_2) \to \alpha \in w''$
iff $\forall w'', \langle w, w'' \rangle \in \Lambda_{\mathcal{C}}(u_1) \circ \Lambda_{\mathcal{C}}(u_2) \to \alpha \in w''$
iff $\forall w'', \langle w, w'' \rangle \in \Lambda_{\mathcal{C}}(u_1 \triangleright u_2) \to \alpha \in w''$.

[If Part] We suppose that $\mathtt{OB}_u\alpha \notin w$. Thus, $\neg\mathtt{OB}_u\alpha \in w$, and by the definition of $w{\downarrow}^u_\alpha$, $\neg\alpha \in w{\downarrow}^u_\alpha$, and since $w{\downarrow}^u_\alpha \subseteq (w{\downarrow}^u_\alpha)^*$, thus $\neg\alpha \in (w{\downarrow}^u_\alpha)^*$ and $\alpha \notin (w{\downarrow}^u_\alpha)^*$. Also, by the definition of $W_{\mathcal{C}}$, $(w{\downarrow}^u_\alpha)^* \in W_{\mathcal{C}}$ and by the definition of $w{\downarrow}^u_\alpha$, for all formulae β, if $\mathtt{OB}_u\beta \in w$, then $\beta \in (w{\downarrow}^u_\alpha)^*$. Thus, by the definition of $\Lambda_{\mathcal{C}}$, $\langle w, (w{\downarrow}^u_\alpha)^* \rangle \in \Lambda_{\mathcal{C}}(u)$. Hence, there exists a world $w' = (w{\downarrow}^u_\alpha)^* \in W_{\mathcal{C}}$ such that $\langle w, w' \rangle \in \Lambda_{\mathcal{C}}(u)$ and $\alpha \notin w'$, which contradicts the hypothesis. ■

The proofs of the other lemmas in MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$ are similar to the ones presented for Core MA(DL)^2. Hence, we have the following theorem.

Theorem 5.2.2 (Strong Completeness of MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$) *MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$ is strongly complete, i.e., if $\Gamma \vDash \alpha$, then $\Gamma \vdash \alpha$*

Proof. Since the model existence lemma holds, we conclude that strong completeness holds as well; if $\Gamma \vDash \alpha$, then $\Gamma \vdash \alpha$. ■

5.2.3 Completeness of MA(DL)$^2[^{-}_{\mathcal{D}}]$ Logic

The completeness proof of the MA(DL)$^2[^{-}_{\mathcal{D}}]$ logic is similar to the completeness proofs of Core MA(DL)^2 and MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^{-}}$. Thus, we do not mention the details of the completeness proof of the MA(DL)$^2[^{-}_{\mathcal{D}}]$ logic, and just concentrate on the parts that have significant different. To this aim, at first, we should define $\Psi{\downarrow}^{u@d}_{\alpha}$ instead of $\Psi{\downarrow}^{u}_{\alpha}$.

Definition 5.2.5 ($\Psi{\downarrow}^{\mathbf{u@d}}_{\alpha}$ Notation) *Let u be an authority, d be a security domain, α be a formula, and Ψ be a maximal consistent set such that $\neg \mathit{OB}_{u@d}\alpha \in \Psi$. We define a set $\Psi{\downarrow}^{u@d}_{\alpha}$ of formulae as follows:*

$$\Psi{\downarrow}^{u@d}_{\alpha} = \{\beta | \mathit{OB}_{u@d}\beta \in \Psi\} \cup \{\neg\alpha\}$$

By the above definition, Lemma 5.2.3 is stated by the new notation and proved similar to the previous one. Now, we can define the canonical model in this logic. For this purpose, we first define the following partitioning for the equivalent security domains.

$$\hat{\mathcal{D}}_{\Gamma} = \{[d]_{\Gamma} |\ d \in \mathcal{D} \wedge\ d \text{ exists in } \Gamma\}$$
$$[d]_{\Gamma} = \{d' |\ d \preceq d', d' \preceq d \in \Gamma^* \wedge\ d, d' \in \mathcal{D}\}$$

Definition 5.2.6 (Canonical Model in MA(DL)$^2[^{-}_{\mathcal{D}}]$) *Let Γ be a consistent set. By Lemma 5.2.1, there exists a maximal consistent set Γ^* such that $\Gamma \subseteq \Gamma^*$. The canonical model generated by Γ is a Kripke structure $\mathcal{CM} = \langle W_{\mathcal{C}}, \Lambda_{\mathcal{C}}, \Delta_{\mathcal{C}}, I_{\mathcal{C}}, \Phi_{\mathcal{C}}, \mathbb{D}_{\mathcal{C}}, J_{\mathcal{C}}\rangle$, where all elements except $\Lambda_{\mathcal{C}}$ and $\mathbb{D}_{\mathcal{C}}$, and $J_{\mathcal{C}}$ are defined similar to the ones defined in Definition 5.2.3 and $\mathbb{D}_{\mathcal{C}}$ and $J_{\mathcal{C}}$ are defined as follows:*

- *$\Lambda_{\mathcal{C}}$ is a function such that for each $u \in \mathcal{U}$ and for each $w_i, w_j \in W_{\mathcal{C}}$, we have $\langle w_i, w_j\rangle \in \Lambda_{\mathcal{C}}(u)$ iff for each formula α, if $\mathit{OB}_{u@d}\alpha \in w_i$ and $x \in J(d)$, then $\alpha \in w_j$.*
- *$\mathbb{D}_{\mathcal{C}}$ is the smallest set such that for each $[d]_{\Gamma} \in \hat{\mathcal{D}}_{\Gamma}$, contains a unique element x.*
- *$J_{\mathcal{C}}$ for all security domains d_j existing in $[d]_{\Gamma}$ is defined as $J_{\mathcal{C}}(d_j) = \bigcup\limits_{d_i \preceq d_j \in \Gamma^*} J_{\mathcal{C}}(d_i) \cup \{x\}$.*

We have lemmas in the MA(DL)$^2[^{-}_{\mathcal{D}}]$ logic similar to the ones mentioned for the Core MA(DL)^2 logic. Moreover, the following lemma holds in this logic.

Lemma 5.2.10 *Let Γ be a consistent set in MA(DL)$^2[^{-}_{\mathcal{D}}]$ logic and $\mathcal{CM} = \langle W_{\mathcal{C}}, \Lambda_{\mathcal{C}}, \Delta_{\mathcal{C}}, I_{\mathcal{C}}, \Phi_{\mathcal{C}}, \mathbb{D}_{\mathcal{C}}, J_{\mathcal{C}}\rangle$ be a canonical model generated by Γ in this logic. The following statements hold in this logic.*

(i) If $\Gamma \vdash d \preceq d'$, then for each $w \in W_{\mathcal{C}}$ we have $d \preceq d' \in w$.

(ii) If there exists a world $w \in W_{\mathcal{C}}$ such that $d \preceq d' \in w$, then $\Gamma \vdash d \preceq d'$.

(iii) For each $d, d' \in \mathcal{D}$, we have $J(d) \subseteq J(d')$ iff $d \preceq d' \in w$.

Proof. It is similar to the proof of Lemma 5.2.5 and Lemma 5.2.6. Note that in this proof, we need to use axiom DSO. ■

The above lemma is leveraged for verifying of the canonical model and proving the coincident lemma in $\text{MA(DL)}^2[^{-}_{\mathcal{D}}]$ logic. Now, we can prove the following theorem.

Theorem 5.2.3 (Strong Completeness of $\text{MA(DL)}^2[^{-}_{\mathcal{D}}]$) *The $\text{MA(DL)}^2[^{-}_{\mathcal{D}}]$ logic is strongly complete, i.e., if $\Gamma \vDash \alpha$, then $\Gamma \vdash \alpha$*

Proof. Considering the definition of canonical model in this logic and holding the model existence lemma, we conclude that strong completeness holds as well; if $\Gamma \vDash \alpha$, then $\Gamma \vdash \alpha$. ■

5.2.4 Completeness of $\text{MA(DL)}^2[^{\mathcal{U}}_{\mathcal{D}}]$ Logic

As mentioned in previous sections, existence of collaboration composition operator (i.e., & operator) in logic of authorities makes problems in the completeness of the $\text{MA(DL)}^2[^{\mathcal{U}}_{-}]$ logic, by eliminating it from this logic, we get a complete logic. Hence, similar to the two previous sections, by integrating the completeness proofs of the $\text{MA(DL)}^2[^{\mathcal{U}}_{-}]_{\&^-}$ and $\text{MA(DL)}^2[^{-}_{\mathcal{D}}]$ logics, we conclude that the $\text{MA(DL)}^2[^{\mathcal{U}}_{\mathcal{D}}]_{\&^-}$ logic is strongly complete.

5.3 Decidability

For employing logics in applications, they are supposed to be decidable. Decidability of a logic has two meanings; decidability of the satisfiability problem, and decidability of the validity problem. Since these problems can be reduced to each other, usually one of them is investigated and then, the result is cascaded to other one. In the rest of this section, we investigate the decidability of the satisfiability problem.

For making MA(DL)^2 decidable, some restrictions are considered in designing it, e.g., variables over concepts are not taken into account, although, it might be useful for expressing some properties in our application. Note that description logic and propositional poly-modal logic, which are the bases of Core MA(DL)^2 are decidable. However, their composition accompanying with adding n-ary predicates on concepts in Core MA(DL)^2, and also adding logic of authorities and logic of security domains do not guarantee the decidability.

In this section, we prove the decidability of the MA(DL)^2 logic family using two approaches; reducing to the satisfiability problem for two-variable first order logic (FO^2), and constructing finite model using filtering method.

5.3.1 Reduction to the Satisfiability Problem for FO^2

One of the approaches to prove the decidability of satisfiability problem for a logic, is to reduce the problem to satisfiability problem for a known decidable logic. For this purpose, a function is defined for translating each formula in the first logic to a formula (or a set of formulae) in the known decidable logic. Then, it is proven that the satisfiability of the formula in the first logic is equivalent to the satisfiability of the translated formula(e) in the known decidable logic.

Since description logic and modal logic are reducible to the satisfiability problem for decidable fragments of first-order logic, we considered the same logic for proving the decidability of the MA(DL)^2 logic family. Three different approaches have been taken for extracting its decidable fragments. Imposing restrictions on quantifier prefix and also on relation and function symbols (see e.g., [69]), imposing restrictions on the number of variables (see the surveys by Grädel *et al.* [70, 71] for different types of two-variable FOL), and imposing restrictions on the forms of quantification (see e.g., [9] for guarded and loosely guarded fragments of FOL).

Scott [144] proved that satisfiability in two-variable first-order logic without equality is decidable (by reducing it to the $\forall^2\exists^*$ Godel class without equality). Then, Mortimer [119] proved that satisfiability in two-variable FOL (denoted by FO^2 or L^2) with equality is decidable, too. To prove the decidability of satisfiability in Core MA(DL)^2, we reduce it to the satisfiability problem for FO^2. For this purpose, we translate formulae of an MA(DL)^2 security knowledge base $\mathcal{K}$ into formulae in FO^2 in polynomial time by the following steps. Then, we show that the satisfiability of formula α in the Core MA(DL)^2 logic is equivalent to the satisfiability of $\exists w.\alpha^\pi(w)$ in FO^2, where $\alpha^\pi(w)$ is the translation of α at world w. Moreover, it is proven that the validity of formula α in Core MA(DL)^2 is equivalent to the validity of $\forall w.\alpha^{\pi}(w)$ in FO^2.

Translation from Core MA(DL)^2 to FO^2

At first, we define an inductive translation function τ as $C \mapsto C^{\tau}(y)$ for atomic and complex concepts in $\mathcal{TB}$ according to the following.

$$\begin{aligned}
&C^{\tau}(y) = \overline{C}(y), \quad \text{if } C \in \mathcal{C} \text{ is an atomic concept}\\
&R^{\tau}(x,y) = \overline{R}(x,y)\\
&(C_1 \sqcap C_2)^{\tau}(y) = {C_1}^{\tau}(y) \wedge {C_2}^{\tau}(y)\\
&(C_1 \sqcup C_2)^{\tau}(y) = {C_1}^{\tau}(y) \vee {C_2}^{\tau}(y)\\
&(\forall R.C)^{\tau}(y) = \forall x.(R^{\tau}(y,x) \rightarrow C^{\tau}(x))\\
&(\exists R.C)^{\tau}(y) = \exists x.(R^{\tau}(y,x) \wedge C^{\tau}(x))\\
&(\neg C)^{\tau}(y) = \neg C^{\tau}(y)
\end{aligned}$$

In fact, the set of objects in Δ satisfying $\overline{C}(y)$ construct the $[\![C]\!]^{I}_{w}$ set which is the interpretation of concept C independent of the possible worlds. Similarly,the set of binary tuples in $\Delta \times \Delta$ satisfying $\overline{R}(x,y)$ construct the $[\![R]\!]^{I}_{w}$ set, which is the interpretation of role R independent of the world w. Note that if we did not have the independency of the interpretation of concepts and roles from the possible worlds, we could not prove the decidability of the Core MA(DL)^2 logic by reducing it to FO^2.

We define another inductive translation function π as $\alpha \mapsto \alpha^{\pi}(x)$ for formula α in Core MA(DL)^2 according to the following. Note that, the variable x in the translation can be considered as a representative of a possible world.

$$(p_i)^\pi(x) = \overline{p_i}(x), \quad \text{if } p_i \in \mathcal{P}_0 \text{ is a proposition}$$
$$(\mathtt{T})^\pi(x) = True \quad (\mathtt{F})^\pi(x) = False$$
$$(\alpha_1 \wedge \alpha_2)^\pi(x) = ({\alpha_1}^\pi(x) \wedge {\alpha_2}^\pi(x))$$
$$(\alpha_1 \vee \alpha_2)^\pi(x) = ({\alpha_1}^\pi(x) \vee {\alpha_2}^\pi(x))$$
$$(\alpha_1 \rightarrow \alpha_2)^\pi(x) = ({\alpha_1}^\pi(x) \rightarrow {\alpha_2}^\pi(x))$$
$$(\neg\alpha)^\pi(x) = \neg\alpha^\pi(x)$$
$$(C_1 \sqsubseteq C_2)^\pi(x) = \forall y.({C_1}^\tau(y) \rightarrow {C_2}^\tau(y))$$
$$(C_1 \equiv C_2)^\pi(x) = (C_1 \sqsubseteq C_2)^\pi(x) \wedge (C_2 \sqsubseteq C_1)^\pi(x) = \forall y.({C_1}^\tau(y) \leftrightarrow {C_2}^\tau(y))$$

$$(p(C_1, ..., C_n))^\pi(x) = \forall y.({C_1}^\tau(y) \rightarrow \overline{p}_{(-,C_2,...,C_n)}(y,x)) \wedge ... \wedge \forall y.({C_n}^\tau(y) \rightarrow \overline{p}_{(C_1,...,C_{n-1},-)}(y,x))$$
$$(\mathtt{OB}_u\alpha)^\pi(x) = \forall y.({E_u}^\rho(x,y) \rightarrow \alpha^\pi(y))$$
$$(\mathtt{PE}_u\alpha)^\pi(x) = \exists y.({E_u}^\rho(x,y) \wedge \alpha^\pi(y))$$
$$(\mathtt{IM}_u\alpha)^\pi(x) = \forall y.({E_u}^\rho(x,y) \rightarrow \neg\alpha^\pi(y))$$
$$(C(a))^\pi(x) = C^\tau(\overline{a})$$
$$(R(a,b))^\pi(x) = R^\tau(\overline{a},\overline{b})$$
$${E_u}^\rho(x,y) = \overline{E_u}(x,y), \text{ if } u \in \mathcal{U} \text{ is a primitive authority}$$

In translation π, the following gists should be considered:

- In translation of t-subformulae $C_1 \sqsubseteq C_2$ and $C_1 \equiv C_2$, the variable x on the left is not appeared in the translation. This is resulted from the assumption about the ontology (as a specification of a conceptualization[73]) where the interpretation of concepts and roles in different possible worlds are the same and thus their translation is independent of variable x. Having such an assumption and removing variable x in this step, prevents increasing the number of variables in the translation of predicates on concepts and also modal formulae defined over them.

- Each n-ary predicate on concepts is converted to a set of binary predicates with two variables. Due to the subsumption relation between the concepts occurring in predicates, after translating predicates on concepts to binary ones in FO^2, we must append certain formulae (resulted from the semantics of concepts) to represent the relationship between the binary predicates with the same name but different indexes. For example, the π translation of $p(C_1, C_2)$ results in appearing a predicate $p_{(-,C_2)}(y,x)$ and $p_{(C_1,-)}(y,x)$. Existence of $C_3 \sqsubseteq C_1$ results in having $p_{(C_1,-)}(y,x) \rightarrow p_{(C_3,-)}(y,x)$ semantically. Thus, we require to append such relationships in a separate step as follows. For each subformula[2] $\forall y.(\overline{C}_1(y) \rightarrow \overline{C}_2(y))$ and predicate $\overline{p}_{(\ldots,C_1,\ldots)}(y,x)$ or $\overline{p}_{(\ldots,C_2,\ldots)}(y,x)$, produced during the above translation step, the following formula must be appended. $\forall y.(\overline{C}_1(y) \rightarrow \overline{C}_2(y)) \rightarrow \forall y.(\overline{p}_{(\ldots,C_2,\ldots)}(y,x) \rightarrow \overline{p}_{(\ldots,C_1,\ldots)}(y,x))$

- Binary predicate ${E_u}^{\rho}(x,y)$ shows the binary relation between the possible world from authority u's viewpoint. ρ is a translation function that depends on authority u. Since in Core MA(DL)^2, only primitive (atomic) authorities exist, ${E_u}^{\rho}(x,y)$ is translated to $\overline{E_u}(x,y)$. Definition of ρ for composite (non-atomic) authorities is presented in next section, when describing the decidability of MA(DL)$^2[^{\mathcal{U}}_{-}]$.

After translating formulae in Core MA(DL)^2 to formulae in FO^2, we discuss the relationship between the satisfying model in Core MA(DL)^2 and satisfying model in FO^2, and also the relationships between the satisfiability problem for Core MA(DL)^2 and FO^2. The remained gist that should be considered in this stage is the effect of seriality property of relations $\Lambda(u)$ (and other properties which are required in the extensions of the Core MA(DL)^2 logic) on the reduction process. As mentioned in the rest, this property (and other similar ones) are taken into account by adding some conditional subformulae. For example, for seriality, subformula $\bigwedge_{u\in\mathcal{U}} \forall x.\exists y.\overline{E_u}(x,y)$ is added to the translated formula conditionally. In this way, a formula α is satisfiable in MA(DL)^2 if $\bigwedge_{u\in\mathcal{U}} \forall x.\exists y.\overline{E_u}(x,y) \rightarrow (\exists x.\alpha^{\pi}(x))$ is satisfiable in FO^2.

Definition 5.3.1 *Let $\mathcal{M} = \langle W, \Lambda, \Delta, I, \Phi\rangle$ be a Kripke model in Core MA(DL)^2. which satisfies formula α. In other words, there exist a world $w \in W$, such that $\models^{\mathcal{M}}_{w} \alpha$. We construct a model $\mathcal{M}^{\pi}$ in FO^2 and prove that $\mathcal{M}$ satisfies α if and only if $\mathcal{M}^{\pi}$ satisfies $\exists w.\alpha^{\pi}(w)$. $\mathcal{M}^{\dagger} = \langle \mathfrak{D}^{\dagger}, \mathfrak{I}^{\dagger}\rangle$, where $\mathfrak{D}^{\dagger}$ is a universe of discourse, $\mathfrak{I}^{\dagger}$ is an interpretation function of propositions, predicates, and terms (names of individuals). They are defined*

[2]Note that C_1 and C_2 in predicate p are atomic and hence ${C_1}^{\tau} = \overline{C}_1$ and ${C_2}^{\tau} = \overline{C}_2$

formally as follows:

$$\begin{aligned}
&\mathfrak{D}^\dagger = W \cup \Delta\\
&\mathfrak{I}^\dagger(\overline{p}_i) = \Phi(p_i), p_i \textit{ is a proposition}\\
&\mathfrak{I}^\dagger(\overline{E}_{u_i}) = \Lambda(u_i)\\
&\mathfrak{I}^\dagger(\overline{p}_{(C_1,...,C_{i-1},-,C_{i+1},...,C_n)}) = \bigcup_{w\in W}\{\langle a_i, w\rangle |\\
&\qquad\qquad \textit{for all } a_1 \in [\![C_1]\!]^I_w, ..., a_{i-1} \in [\![C_{i-1}]\!]^I_w, a_{i+1} \in [\![C_{i+1}]\!]^I_w, ..., a_n \in [\![C_n]\!]^I_w,\\
&\qquad\qquad \langle a_1, ..., a_{i-1}, a_i, a_{i+1}..., a_n\rangle \in [\![p]\!]^I_w\}\\
&\mathfrak{I}^\dagger(\overline{C}_i) = [\![C_i]\!]^I_w, \textit{for an arbitrary } w\\
&\mathfrak{I}^\dagger(\overline{R}_i) = [\![R_i]\!]^I_w, \textit{for an arbitrary } w\\
&\mathfrak{I}^\dagger(\overline{a}_i) = [\![a_i]\!]^I_w, \textit{for an arbitrary } w
\end{aligned}$$

The interpretation of each predicate $\overline{p}_i$, which is produced from the π translation of proposition p_i, is the set of possible worlds in model $\mathcal{M}$, where p_i is true. The interpretation of each predicate $\overline{E}_{u_i}$ in the above definition, is the binary relation $\Lambda(u_I)$ on possible worlds W, which is a subset of universe of discourse $\mathfrak{D}^\dagger$. The interpretation of each predicate $\overline{p}_{C_1,..,C_{i-1},-,C_{i+1},...,C_n}$ is the union of vertical projection of interpretation of predicate p in model $\mathcal{M}$ in worlds w in combination with w. The interpretation of predicate $\overline{C}_i$ is the set of objects in the interpretation of C_i. Note that since the interpretation of concepts in different worlds is the same, for all $w \in W$ in model $\mathcal{M}$ the sets $[\![C_i]\!]^I_w$ are the same and they are independent of the worlds w. The similar justifications exist for the interpretation of $\overline{R}_i$ and $\overline{a}_i$.

Proposition 5.3.1 *Let α be a formula in Core MA(DL)^2. α is satisfiable in model $\mathcal{M}$ of Core MA(DL)^2 logic if and only if $\exists w.\alpha^\pi(w)$ is satisfiable in model $\mathcal{M}^\dagger$ of FO^2 logic.*

Proof. We prove the proposition by induction on the length of formula α.
Basis: α is an atomic formula. We prove for $p(C_1, ..., C_n)$ and $C_1 \sqsubseteq C_2$ as an example. Other cases including $C_1 \equiv C_2$, $C(a)$, $R(a, b)$, and p_i (as a proposition) are easily proved following the definition of $\mathcal{M}^\dagger$.

- $\alpha = p(C_1, ..., C_n)$, an n-ary predicate.
  $\mathcal{M}$ satisfies $p(C_1, ..., C_n)$

iff there exists a world w, $\models_w^{\mathcal{M}} p(C_1, ..., C_n)$
iff there exists w, such that $w \in \Phi(p(C_1, ..., C_n))$ and thus $\prod_{i=1}^{n} [\![C_i]\!]_w^I \subseteq [\![p]\!]_w^I$
iff there exists w, such that
(for all a_1, (if $a_1 \in [\![C_1]\!]_w^I$, then for all $a_2 \in [\![C_2]\!]_w^I, ..., a_n \in [\![C_n]\!]_w^I, \langle a_1, ..., a_n\rangle \in [\![p]\!]_w^I$)
and ... and
(for all a_n, (if $a_n \in [\![C_n]\!]_w^I$, then for all $a_1 \in [\![C_1]\!]_w^I, ..., a_{n-1} \in [\![C_{n-1}]\!]_w^I, \langle a_1, ..., a_n\rangle \in$ $[\![p]\!]_w^I$)
iff there exists w, such that
(for all a_1, if $a_1 \in \mathfrak{I}^\dagger(\overline{C}_1)$, then $\langle a_1, w\rangle \in \mathfrak{I}^\dagger(\overline{p}_{(-,C_2,...,C_n)})$), and ... and
(for all a_n, if $a_n \in \mathfrak{I}^\dagger(\overline{C}_n)$, then $\langle a_n, w\rangle \in \mathfrak{I}^\dagger(\overline{p}_{(C_1,...,C_{n-1},-)})$)
iff $\models^{\mathcal{M}^\dagger} \exists w.[\forall y.({C_1}^\tau(y) \to \overline{p}_{(-,C_2,...,C_n)}(y, w)) \wedge ... \wedge \forall y.({C_n}^\tau(y) \to \overline{p}_{(C_1,...,C_{n-1},-)}(y, w))]$
iff $\models^{\mathcal{M}^\dagger} \exists w.(p(C_1, ..., C_n))^\pi(w)$

- $\alpha = C_1 \sqsubseteq C_2$, a subsumption relationship. Without scarifying generality, we suppose that C_1 and C_2 are atomic concepts (i.e., $C_1, C_2 \in \mathcal{C}$).
$\mathcal{M}$ satisfies $C_1 \sqsubseteq C_2$
iff there exists a world w, such that $\models_w^{\mathcal{M}} C_1 \sqsubseteq C_2$
iff there exists w, such that $w \in \Phi(C_1 \sqsubseteq C_2)$ and thus $[\![C_1]\!]_w^I \subseteq [\![C_2]\!]_w^I$
iff there exists w, such that for all y, if $y \in [\![C_1]\!]_w^I$, then $y \in [\![C_2]\!]_w^I$
iff there exists w, such that for all y, if $y \in \mathfrak{I}^\dagger(\overline{C}_1)$, then $y \in \mathfrak{I}^\dagger(\overline{C}_2)$
iff $\models^{\mathcal{M}^\dagger} \exists w.(\forall y.{C_1}^\tau(y) \to {C_2}^\tau(y))$ iff $\models^{\mathcal{M}^\pi} \exists w.(C_1 \sqsubseteq C_2)^\pi(w)$

Induction Step: we suppose that the proposition holds for β, and prove for $\alpha = \mathtt{OB}_u\beta$ as a sample. Other cases including $\alpha = \alpha_1 * \alpha_2$ ($*$ is $\wedge, \vee$, or $\to$), $\alpha = \neg\beta$, $\alpha = \mathtt{PE}_u\beta$, and $\alpha = \mathtt{IM}_u\beta$ are proved similarly.

- $\mathcal{M}$ satisfies $\alpha = \mathtt{OB}_u\beta$
iff by proposition 4.2.1, there exists a world w, $\models_w^{\mathcal{M}} \mathtt{OB}_u\beta$
iff there exists w, for all w', if $\langle w, w'\rangle \in \Lambda(u)$, then $\models_{w'}^{\mathcal{M}} \beta$
iff there exists w, for all w', if $\langle w, w'\rangle \in \mathfrak{I}^\dagger(\overline{E}_u)$, then $\models^{\mathcal{M}^\pi} \beta^\pi(w')$
iff $\models^{\mathcal{M}^\dagger} \exists w.(\forall w'.\overline{E}_u(w, w') \to \beta^\pi(w'))$ iff $\models^{\mathcal{M}^\dagger} \exists w.(\mathtt{OB}_u\beta)^\pi(w)$

■

Definition 5.3.2 *Let $\mathcal{N} = \langle \mathfrak{D}, \mathfrak{I} \rangle$ be a model in FO^2 which satisfies the conditional formula $\forall.\exists.y.\overline{E}_u(x,y)$ for all u_i in the language. A Kripke model $\mathcal{N}^\sharp \langle W^\sharp, \Lambda^\sharp, \Delta^\sharp, I^\sharp, \Phi^\sharp \rangle$ in Core MA(DL)^2 is defined as follows.*

$W^\sharp \subseteq \mathfrak{D}$

$\Delta^\sharp \subseteq \mathfrak{D}$

$\Phi^\sharp(p_i) = \mathfrak{I}(\overline{p}_i), p_i$ *is a proposition*

$\Lambda^\sharp(u_i) = \mathfrak{I}(\overline{E}_{u_i})$

$[\![p]\!]_w^{I^\sharp} = \{\langle a_1, ..., a_n \rangle |\ \langle a_1, w \rangle \in \mathfrak{I}(\overline{p}_{(-,C_2,...,C_n)}) \wedge ... \wedge \langle a_n, w \rangle \in \mathfrak{I}(\overline{p}_{(C_1,...,C_{n-1},-)}))\},$ *for all* $w \in W$

$[\![C_i]\!]_w^{I^\sharp} = \mathfrak{I}(\overline{C}_i),$ *for all* $w \in W^\sharp$

$[\![R_i]\!]_w^{I^\sharp} = \mathfrak{I}(\overline{R}_i),$ *for all* $w \in W^\sharp$

$[\![a_i]\!]_w^{I^\sharp} = \mathfrak{I}(\overline{a}_i),$ *for all* $w \in W^\sharp$

Proposition 5.3.2 *Let α be a formula in Core MA(DL)^2. If $\mathcal{N}$ is a model of FO^2 such that $\models^{\mathcal{N}} \forall x.\exists y.\overline{E}_{u_i}(x,y)$ for all authorities u_i existing in formula α, then α in model $\mathcal{N}^\sharp$ of Core MA(DL)^2 is satisfiable if and only if $\exists w.\alpha^\pi(w)$ is satisfiable in model $\mathcal{N}$ of FO^2.*

Proof. Satisfying the condition $\forall x.\exists y.\overline{E}_{u_i}(x,y)$ for each authority u_i concludes that $\mathcal{N}^\sharp$ is a valid model for Core MA(DL)^2 where the seriality of relations $\Lambda(u_i)$ hold. We can easily prove the proposition by induction on the structure of formula α, analogous to the proof of proposition 5.3.1. ■

Definition 5.3.3 (Size of Formula) *Size of a formula is equal to the number of nodes in its parse tree, i.e., the number of occurrences of logical connectives, quantifiers, modal operators, operators, and atomic symbols.*

More precise, size of a formula in Core MA(DL)^2 is calculated inductively as follows:

$$
\begin{array}{lr}
|C_1 \sqsubseteq C_2| = |C_1 \equiv C_2| = |C_1| + |C_2| + 1 & \\
|C_1 \sqcup C_2| = |C_1 \sqcap C_2| = |C_1| + |C_2| + 1 & \\
|\forall R.C| = |\exists R.C| = |R| + |C| + 1 & \\
|\neg C| = |C| + 1 & \\
|C| = 1 & C \in \mathcal{C} \\
|R| = 1 & R \in \mathcal{R} \\
|C(a)| = |C| + 1 & a \in \mathcal{I} \\
|R(a,b)| = |R| + 2 & a, b \in \mathcal{I} \\
|p| = |\mathtt{T}| = |\mathtt{F}| = 1 & p \text{ is a proposition} \\
|\neg\alpha| = |\alpha| + 1 & \\
|p(C_1, ..., C_n)| = \sum_{i=1}^{n} |C_i| + 1 & \\
|\alpha_1 * \alpha_2| = |\alpha_1| + |\alpha_2| + 1 & * \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \\
|\mathtt{ds}_u \alpha| = |\alpha| + |u| + 1 & \mathtt{ds} \in \{\mathtt{OB}, \mathtt{PE}, \mathtt{IM}, \mathtt{GR}\} \\
|u| = 1 & u \in \mathcal{U}
\end{array}
$$

Following the above definition, the size of an SKB is the aggregation of the size of its formulae.

Lemma 5.3.1 *Let $\mathcal{K}$ be an MA(DL)^2 SKB and $\mathcal{K}^\pi$ be the result of π translation of $\mathcal{K}$ in FO^2. Then, we have*

(i) $\mathcal{K}^\pi$ *has size* $O(|\mathcal{K}|)$.

(ii) $\mathcal{K}^\pi$ *is constructed from* $\mathcal{K}$ *in polynomial time.*

(iii) $\mathcal{K} \vDash^{\mathcal{M}} \alpha$ *holds in Core MA(DL)^2 if and only if* $\forall w.\mathcal{K}^\pi(w) \vDash^{\mathcal{M}^\dagger} \exists w.\alpha^\pi(w)$ *holds in FO^2.*

(iv) $\bigwedge_{u_i} \forall x.\exists y.\overline{E}_{u_i}(x,y) \wedge \forall w.\mathcal{K}^\pi(w) \vDash^{\mathcal{N}} \exists w.\alpha^\pi(w)$ *holds in FO^2 if and only if* $\mathcal{K} \vDash^{\mathcal{N}^\sharp}$ *holds in Core MA(DL)^2.*

Proof.

(i) In translation of formulae of $\mathcal{K}$, based on the definition of translation functions and also definition 5.3.3 of the size of formula, the resulted formulae has a size of linear of $|\mathcal{K}|$. The formulae added to $\mathcal{K}^\pi$ (due to the properties of Kripke structure) keeps the size of linear of $|\mathcal{K}|$, because:

1. For each $\overline{E}_u$ exists in $\mathcal{K}^\pi$, we add $\forall x.\exists y.\overline{E}_u(x,y)$. The number of them is at most $|\mathcal{K}|$.
2. For each translated subsumption relationship (e.g., $\forall y.\overline{C}_1(y) \to C_2{}^\tau(y)$) and every related binary predicate (e.g., $\overline{p}_{(\dots,C_1,\dots)}(x,y)$) resulted from n-ary predicates on concepts, an aforementioned rule with constant size is added to $\mathcal{K}^\pi$. Since, the number of required rules are less than $|\mathcal{K}|$, $|\mathcal{K}^\pi|$ remains linear order of $|\mathcal{K}|$.

(ii) The proof of item (i) shows that π translation of formulae in $\mathcal{K}$ takes $O(|\mathcal{K}|)$ time. Adding extra formulae resulted from the properties of Kripke structure semantics, takes $O(|\mathcal{K}|^2)$, because in the second case (specified above), searching for the binary predicates for each formula resulted from the translation of subsumption relationships, takes $O(|\mathcal{K}|^2)$. Hence, the time complexity of translation $\mathcal{K}$ to $\mathcal{K}^\pi$ takes $O(|\mathcal{K}|^2)$ that is a polynomial order of $|\mathcal{K}|$.

(iii) $\mathcal{K} \models^{\mathcal{M}} \alpha$ is equal to $\models^{\mathcal{M}} \mathcal{K} \to \alpha$. By Lemma 5.3.1 we have $\models^{\mathcal{M}} \mathcal{K} \to \alpha$ iff $\models^{\mathcal{M}^\dagger} \exists w.(\mathcal{K} \to \alpha)^\pi(w)$ or in other words $\models^{\mathcal{M}^\dagger} \forall w.\mathcal{K}^\pi(w) \to \exists w.\alpha^\pi(w)$, which holds iff $\forall w.\mathcal{K}^\pi(w) \models^{\mathcal{M}^\dagger} \exists w.\alpha^\pi(w)$.

(iv) By proposition 5.3.2, we can easily prove analogous to the proof of (iii).

■

Theorem 5.3.1 (Decidability of Core MA(DL)^2) *The satisfiability problem for the Core MA(DL)^2 logic is decidable.*

Proof. By items (ii) to (iv) of Lemma 5.3.1, the satisfiability problem for Core MA(DL)^2 is reducible to the satisfiability problem for FO^2 in polynomial time. Since FO^2 is decidable [71], it is obvious that the satisfiability problem for the Core MA(DL)^2 logic (with empty or non-empty SKB) is decidable as well. This concludes that validity problem for this logic is decidable, too. ■

5.3.2 Decidability of MA(DL)$^2[^{\mathcal{U}}_{-}]$ by Reduction

Supplementing MA(DL)^2 with logic of composite authorities, enables us to have modal formulae of the form $\mathtt{ds}_u\alpha$ where u is a composite authority. Thus, to reduce the satisfiability problem from MA(DL)$^2[^{\mathcal{U}}_{-}]$ to FO^2, we require that the translation function ρ (used in definition of π translation function) to be defined as follows.

$$\begin{aligned}
&(\mathsf{OB}_u\alpha)^\pi(x) = \forall y.({E_u}^\rho(x,y) \to \alpha^\pi(y))\\
&{E_u}^\rho(x,y) = \overline{E_u}(x,y),\ \text{if } u \in \mathcal{U} \text{ is a primitive authority}\\
&{E_{(u_1|u_2)}}^\rho(x,y) = {E_{u_1}}^\rho(x,y) \vee {E_{u_2}}^\rho(x,y)\\
&{E_{(u_1\&u_2)}}^\rho(x,y) = {E_{u_1}}^\rho(x,y) \wedge {E_{u_2}}^\rho(x,y)\\
&{E_{(u_1\rhd u_2)}}^\rho(x,y) = \exists z.\,({E_{u_1}}^\rho(x,z) \wedge {E_{u_2}}^\rho(z,y))
\end{aligned}$$

In the above definition, translation of $(u_1 \rhd u_2)$ requires three variables. This is resulted from the fact that composition of two relations (i.e., $\Lambda(u_1) \circ \Lambda(u_2)$) is not definable from relations (i.e., $\Lambda(u_1)$ and $\Lambda(u_2)$) in FO^2 [71]. However, in our case we can limit the number of variables to two, by rewriting the translation of $(\mathsf{OB}_{(u_1\rhd u_2)}\alpha)^\pi(x)$ as follows:

$$\begin{aligned}
&(\mathsf{OB}_{(u_1\rhd u_2)}\alpha)^\pi(x) =\\
&\qquad \forall y.((\exists z.{E_{u_1}}^\rho(x,z) \wedge {E_{u_2}}^\rho(z,y)) \to \alpha^\pi(y)) =\\
&\qquad\qquad \forall z.({E_{u_1}}^\rho(x,z) \to \forall y.({E_{u_2}}^\rho(z,y) \to \alpha^\pi(y)))
\end{aligned}$$

By replacing z with y and concurrently reusing variable x instead of y (because x is not free or bound in the subformula) in the above formula we obtain the following:

$$(\mathsf{OB}_{(u_1\rhd u_2)}\alpha)^\pi(x) = \forall y.({E_{u_1}}^\rho(x,y) \to \forall x.({E_{u_2}}^\rho(y,x) \to \alpha^\pi(x)))$$

Following the above discussion, we require to convert the π translation of $\mathsf{OB}_{(u_1|u_2)}\alpha$ and $\mathsf{OB}_{(u_1\&u_2)}\alpha$ to the combination of formulae in canonical form $\forall y.({E_u}^\rho(x,y) \to \beta^\pi(y))$. It is convenient to show that the π translation of $\mathsf{OB}_{(u_1|u_2)}\alpha$ can be written in the canonical form as follows:

$$(\mathsf{OB}_{(u_1|u_2)}\alpha)^\pi(x) = (\forall y.({E_{u_1}}^\rho(x,y) \to \alpha^\pi(y))) \wedge (\forall y.({E_{u_2}}^\rho(x,y) \to \alpha^\pi(y)))$$

However, we cannot convert the π translation of $\mathtt{OB}_{(u_1\&u_2)}\alpha$ to the canonical form. Thus, we cannot translate the composition of $\triangleright$ and & into FO^2 because we require at least three variables for this purpose. Note that, this does not mean that their composition makes the logic undecidable, because there might exist other approaches to prove its decidability (e.g., we might prove that it has the finite model property and hence it is decidable).

Following the above discussion, we cannot prove the decidability of MA(DL)$^2[^{\mathcal{U}}_{-}]$ using the reduction approach; however, we can prove the decidability of a restricted version of this logic, where the composition of $\triangleright$ and & operators in defining composite authorities is not allowed.

Theorem 5.3.2 (Decidability of Restricted MA(DL)$^2[^{\mathcal{U}}_{-}]$) *The satisfiability problem for restricted version of MA(DL)$^2[^{\mathcal{U}}_{-}]$, where the composition of $\triangleright$ and & operators in defining composite authorities is not allowed, is decidable.*

Proof. By extending the definition of 5.3.3 by the following equation, the definition of the size of formula in MA(DL)$^2[^{\mathcal{U}}_{-}]$ is obtained.

$$|u_1 \# u_2| = |u_1| + |u_2| + 1, \quad \# \in \{\&, |, \triangleright\}$$

By removing the seriality condition on relations between the possible worlds, and proving propositions 5.3.1 and 5.3.2, and also proving the items (i) to (iv) of Lemma 5.3.1 in restricted MA(DL)$^2[^{\mathcal{U}}_{-}]$, we can conclude that this logic is decidable. ■

By the discussion presented in Section 4.3.5, the seriality condition cannot be preserved in collaborative composition (by & operator). This operator also makes the presented proof theory incomplete. Thus, in practice it is better to remove this operator to have a complete and decidable logic; based on theorem 5.2.2 and the following decidability theorem.

Theorem 5.3.3 (Decidability of MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$) *The satisfiability problem for MA(DL)$^2[^{\mathcal{U}}_{-}]_{\&^-}$ is decidable.*

Proof. It is similar to the proof of theorem 5.3.2. ■

5.3.3 Decidability of MA(DL)$^2[^{-}_{\mathcal{D}}]$ by Reduction

For translating the formulae in the MA(DL)$^2[^{-}_{\mathcal{D}}]$ logic, we just require to change the definition of translation function π for the modal formulae as follows.

$$(\mathtt{OB}_{u@d}\alpha)^\pi(x) = \forall z.\overline{d}(z) \to (\forall y.({E_u}^\rho(z,x,y) \to \alpha^\pi(y)))$$

Since $\mathbb{D}$ is finite and the domain of variable z is limited to set $\mathbb{D}$. Thus, we can eliminate variable z and replace z with the elements of $\mathbb{D}$. In this way, we can replace each predicate $E_u(z,x,y)$ with predicates of the form $E_{u,z}(x,y)$ with two variables, and also replace each predicate $\overline{d}(z)$ with a set of propositions $\overline{d}_z$. Regarding the above discussion, the π translation of the modal formulae is as follows.

$$\begin{aligned}
(\mathtt{OB}_{u@d}\alpha)^\pi(x) &= \bigwedge_{z\in\mathbb{D}} \forall y.(\overline{d}_z \to ({E_{u,z}}^\rho(x,y) \to \alpha^\pi(y))) \\
(\mathtt{PE}_{u@d}\alpha)^\pi(x) &= \bigvee_{z\in\mathbb{D}} \exists y.(\overline{d}_z \wedge ({E_{u,z}}^\rho(x,y) \wedge \alpha^\pi(y)) \\
(\mathtt{IM}_{u@d}\alpha)^\pi(x) &= \bigwedge_{z\in\mathbb{D}} \forall y.(\overline{d}_z \to ({E_{u,z}}^\rho(x,y) \to \neg\alpha^\pi(y)))
\end{aligned}$$

The translation of subdomain relationship $d \preceq d'$ is defined as follows:

$$(d \preceq d')^\pi(x) = \bigwedge_{z\in\mathbb{D}} (\overline{d}_z \to \overline{d'}_z)$$

Now, we need to change definitions 5.3.1 and 5.3.2 as follows.

Definition 5.3.4 *Let $\mathcal{M} = \langle W, \Lambda, \Delta, I, \Phi, \mathbb{D}, J\rangle$ be a Kripke model of MA(DL)$^2[^{-}_{\mathcal{D}}]$. FO^2 model $\mathcal{M}^\dagger = \langle \mathfrak{D}^\dagger, \mathfrak{I}^\dagger\rangle$ is defined similar to definition 5.3.1 with the following changes.*

$$\begin{aligned}
&\mathfrak{D}^\dagger = W \cup \Delta \cup \mathbb{D} \\
&\mathfrak{I}^\dagger(\overline{E}_{u,z}) = \Lambda(u)(z) \\
&\mathfrak{I}^\dagger(\overline{d}_z) = \begin{cases} 1, & \textit{if } z \in J(d) \\ 0, & \textit{if } \notin J(d) \end{cases} \qquad d \in \mathcal{D} \textit{ is a security domain}
\end{aligned}$$

Definition 5.3.5 *Let $\mathcal{N} = \langle \mathfrak{D}, \mathfrak{I}\rangle$ be a model of FO^2 that satisfies conditional formula*

$\forall x.\exists y.\overline{E}_u(x,y)$. Definition of model $\mathcal{N}^\sharp = \langle W^\sharp, \Lambda^\sharp, \Delta^\sharp, I^\sharp, \Phi^\sharp, \mathbb{D}^\sharp, J^\sharp \rangle$ of MA(DL)$^2[{}^{-}_{\mathcal{D}}]$ is similar to definition 5.3.2 with the following changes.

$$\begin{aligned}
&\mathbb{D}^\sharp \subseteq \mathfrak{D}\\
&\Lambda^\sharp(u)(z) = \mathfrak{I}(\overline{E}_{u,z})\\
&J^\sharp(d) = \{z |\ \mathfrak{I}(\overline{d}_z) = 1\}
\end{aligned}$$

Theorem 5.3.4 (Decidability of MA(DL)$^2[{}^{-}_{\mathcal{D}}]$) *The satisfiability problem for the MA(DL)$^2[{}^{-}_{\mathcal{D}}]$ logic is decidable.*

Proof. By extending definition 5.3.3 by the following equation, the definition of the size of formula in MA(DL)$^2[{}^{-}_{\mathcal{D}}]$ is obtained.

$$\begin{aligned}
&|\mathtt{ds}_{u@d}\alpha| = |\alpha| + |u| + |d| + 1 && \mathtt{ds} \in \{\mathtt{OB}, \mathtt{PE}, \mathtt{IM}, \mathtt{GR}\}\\
&|d| = 1 && d \in \mathcal{D}
\end{aligned}$$

By proving propositions 5.3.1 and 5.3.2, and also proving the items (i) to (iv) of Lemma 5.3.1 in MA(DL)$^2[{}^{-}_{\mathcal{D}}]$, we can conclude that this logic with the finite security domains is decidable.
■

5.3.4 The Finite Model Property of MA(DL)$^2[{}^{\mathcal{U}}_{\mathcal{D}}]$

A logic has the finite model property (FMP) if and only if every satisfiable formula in the logic is satisfiable in a finite model [29]. If L is finitely axiomatizable and has the finite model property, then it is decidable (Theorem 8.5 in [81]).

To prove that the MA(DL)^2 logic family has the finite model property, we use the *filtration method*, which is the most widely used method to prove this property in modal logics.

In the filtration method, the main idea is constructing the finite model w.r.t a given formula α in the logic (here MA(DL)$^2[{}^{\mathcal{U}}_{\mathcal{D}}]$) in the following manner. Given a formula α in MA(DL)$^2[{}^{\mathcal{U}}_{\mathcal{D}}]$ and a model $\mathcal{M}$ satisfying α, we construct a *finite* model $\mathcal{M}_\alpha$ satisfying α . To this aim, we define the closure of formula α (denoted by $\mathtt{cls}(\alpha)$) as a set of its subformulae. The closure set $\mathtt{cls}(\alpha)$ partitions the set of worlds W in $\mathcal{M}$ to a finite number of equivalent classes in which each class is satisfying the same subset of $\mathtt{cls}(\alpha)$.

Definition 5.3.6 (Closure) *Given a formula α in MA(DL)^2$[^{\mathcal{U}}_{\mathcal{D}}]$, the closure of α, denoted by $\texttt{cls}(\alpha)$, is the smallest set of formulae that is closed under subformula relation.*

In a more precise way, $\texttt{cls}(\alpha)$ in the above definition is obtained by the following rules:

1. $\alpha \in \texttt{cls}(\alpha)$
2. if $\alpha_1 * \alpha_2 \in \texttt{cls}(\alpha)$, then $\alpha_1, \alpha_2 \in \texttt{cls}(\alpha)$; where $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$
3. if $\neg\alpha_1 \in \texttt{cls}(\alpha)$, then $\alpha_1 \in \texttt{cls}(\alpha)$
4. if $\texttt{ds}_{u@d}\alpha_1 \in \texttt{cls}(\alpha)$, where $\texttt{ds} \in \{\texttt{OB}, \texttt{PE}, \texttt{IM}, \texttt{GR}\}$ and $u \in \mathcal{U}^*$, then $\alpha_1 \in \texttt{cls}(\alpha)$

Lemma 5.3.2 *If $\texttt{cls}(\alpha)$ is the closure of formula α, then $\texttt{cls}(\alpha)$ is finite.*

Proof. It is clear that in all the rules introduced for calculating $\texttt{cls}(\alpha)$, the size of the formulae added by the right side is less than the size of formulae on the left side. Since the number of rules are finite, and the size of formula α is finite too, the number of formulae added to set $\texttt{cls}(\alpha)$ is also finite. Thus, set $\texttt{cls}(\alpha)$ is finite. ■

Definition 5.3.7 (Equivalence Relation $\approx_\Gamma$) *Let Γ be a finite set of formulae in MA(DL)^2$[^{\mathcal{U}}_{\mathcal{D}}]$. The equivalence relation $\approx_\Gamma$ on worlds W in model $\mathcal{M}$ is defined as $w \approx_\Gamma w'$ if and only if for all $\alpha \in \Gamma$, we have $\vDash^{\mathcal{M}}_{w} \alpha$ if and only if $\vDash^{\mathcal{M}}_{w'} \alpha$.*

Following the definition of $\approx_\Gamma$, the equivalence class of a world w w.r.t. $\approx_\Gamma$ is defined as $[w]_\Gamma = \{w' \in W | w \approx_\Gamma w'\}$.

Definition 5.3.8 (Filtration of a Model) *Given a model $\mathcal{M} = \langle W, \Lambda, \Delta, I, \Phi, \mathbb{D}, J\rangle$ and a set of formulae Γ in MA(DL)^2$[^{\mathcal{U}}_{\mathcal{D}}]$, a model $\mathcal{M}_\Gamma = \langle W_\Gamma, \Lambda_\Gamma, \Delta_\Gamma, I_\Gamma, \Phi_\Gamma, \mathbb{D}_\Gamma, J_\Gamma\rangle$ is a filtration of $\mathcal{M}$ w.r.t. Γ if for every $\alpha \in \Gamma$ and $w \in W$, we have $\vDash^{\mathcal{M}}_{w} \alpha$ if and only if $\vDash^{\mathcal{M}_\Gamma}_{[w]_\Gamma} \alpha$.*

We define a model $\mathcal{M}_\Gamma$ as follows and prove that it is a filtration of $\mathcal{M}$ w.r.t. Γ.

Definition 5.3.9 *If α is a formula in MA(DL)^2$[^{\mathcal{U}}_{\mathcal{D}}]$ and $\Gamma = \texttt{cls}(\alpha)$, then model $\mathcal{M}_\Gamma = \langle W_\Gamma, \Lambda_\Gamma, \Delta_\Gamma, I_\Gamma, \Phi_\Gamma, \mathbb{D}_\Gamma, J_\Gamma\rangle$ is defined as follows:*

- *$W_\Gamma = \{[w]_\Gamma | w \in W\}$, it is obvious that if Γ is finite, then W_Γ is finite too.*

- $\Lambda_\Gamma(u)(z) = \{\langle [w]_\Gamma, [w']_\Gamma \rangle | \langle w, w' \rangle \in \Lambda(u)(z)\}$, $u \in \mathcal{U}$
- $\Delta_\Gamma = \Delta$
- $I_\Gamma = I$, *i.e.*, $[\![\sigma]\!]^{I_\Gamma}_{[w]_\Gamma} = [\![\sigma]\!]^I_w$, $[\![C]\!]^{I_\Gamma}_{[w]_\Gamma} = [\![C]\!]^I_w$, $[\![R]\!]^{I_\Gamma}_{[w]_\Gamma} = [\![R]\!]^I_w$, $[\![a]\!]^{I_\Gamma}_{[w]_\Gamma} = [\![a]\!]^I_w$, $[\![p]\!]^{I_\Gamma}_{[w]_\Gamma} = [\![p]\!]^I_w$
- $\Phi_\Gamma(p) = \begin{cases} \{[w]_\Gamma | w \in \Phi(p)\} & \textit{, if } p \textit{ is a proposition and } p \in \Gamma; \\ \varnothing & \textit{, if } p \textit{ is a proposition and } p \notin \Gamma. \end{cases}$
- $\mathbb{D}_\Gamma = \mathbb{D}$
- $J_\Gamma = J$

In the above definition, $\Lambda_\Gamma(u)(z)$ is defined for primitive authorities (i.e., $u \in \mathcal{U}$). In the rest, in Lemma 5.3.4, we get the definition of $\Lambda_\Gamma(u)(z)$ for composite authorities (i.e., $u \in \mathcal{U}^*$). Also, interpretation function Φ_Γ is only defined for propositions. The definition of this function for other formulae can be obtained from the definition of other elements of the model (see the proof of Theorem 5.3.5).

Lemma 5.3.3 *For every $C \in \mathcal{C}^*$, we have $[\![C]\!]^I_w = [\![C]\!]^{I_\Gamma}_{[w]_\Gamma}$.*

Proof. It is easy to prove. For example, for $C_1 \sqcap C_2$ and $\forall R.C$, we have

$$[\![C_1 \sqcap C_2]\!]^I_w = [\![C_1]\!]^I_w \cap [\![C_2]\!]^I_w = [\![C_1]\!]^{I_\Gamma}_{[w]_\Gamma} \cap [\![C_2]\!]^{I_\Gamma}_{[w]_\Gamma} = [\![C_1 \sqcap C_2]\!]^{I_\Gamma}_{[w]_\Gamma}$$
$$[\![\forall R.C]\!]^I_w = \{a | \forall b, \langle a, b\rangle \in [\![R]\!]^I_w \to b \in [\![C]\!]^I_w\} = \{a | \forall b, \langle a, b\rangle \in [\![R]\!]^{I_\Gamma}_{[w]_\Gamma} \to b \in [\![C]\!]^{I_\Gamma}_{[w]_\Gamma}\} = [\![\forall R.C]\!]^{I_\Gamma}_{[w]_\Gamma}$$

■

Lemma 5.3.4 *Given $u \in \mathcal{U}^*$, if $\langle w, w'\rangle \in \Lambda(u)(z)$, then $\langle [w]_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u)(z)$.*

Proof. We prove by induction on the structure of composite authority u.
Basis: By the definition of Λ_Γ, the lemma holds for primitive authorities ($u \in \mathcal{U}$).
Induction Step: Suppose that the lemma holds for $u_1, u_2 \in \mathcal{U}^*$. Now, we prove that it holds for different compositions of u_1 and u_2 as follows.

- If $u = (u_1|u_2)$, then $\langle w, w'\rangle \in \Lambda(u_1|u_2)(z)$, iff $\langle w, w'\rangle \in \Lambda(u_1)(z) \cup \Lambda(u_2)(z)$, and thus, $\langle w, w'\rangle \in \Lambda(u_1)(z)$ or $\langle w, w'\rangle \in \Lambda(u_2)(z)$. By the induction assumption, $\langle [w]_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u_1)(z)$ or $\langle [w]_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u_2)(z)$, and hence, $\langle [w]_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u_1)(z) \cup \Lambda_\Gamma(u_2)(z)$. Thus, we have $\langle [w]_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u_1|u_2)(z)$.

- If $u = (u_1 \& u_2)$, the proof is analogous to case $u = (u_1|u_2)$.

- If $u = (u_1 \rhd u_2)$, then $\langle w, w'\rangle \in \Lambda(u_1 \rhd u_2)(z)$ iff $\langle w, w'\rangle \in \Lambda(u_1)(z) \circ \Lambda(u_2)(z)$, and thus there exists $w'' \in W$, $\langle w, w''\rangle \in \Lambda(u_1)(z)$ and $\langle w'', w'\rangle \in \Lambda(u_2)(z)$, implies that there exists $w'' \in W$, $\langle [w]_\Gamma, [w'']_\Gamma\rangle \in \Lambda_\Gamma(u_1)(z)$ and $\langle [w'']_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u_1)(z)$. Thus, there exists $l \in W$, $\langle [w]_\Gamma, l\rangle \in \Lambda_\Gamma(u_1)(z)$ and $\langle l, [w']_\Gamma\rangle \in \Lambda_\Gamma(u_2)(z)$, that concludes $\langle [w]_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u_1)(z) \circ \Lambda_\Gamma(u_2)(z)$, and hence $\langle [w]_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u_1 \rhd u_2)(z)$.

■

Theorem 5.3.5 (Filtration) *If α is a formula in MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ and $\Gamma = \mathtt{cls}(\alpha)$, then model $\mathcal{M}_\Gamma$ (as defined in 5.3.9) is a filtration of model $\mathcal{M}$ w.r.t. Γ.*

Proof. We show for all $\beta \in \Gamma$, $\models^{\mathcal{M}}_w \beta$ if and only if $\models^{\mathcal{M}_\Gamma}_{[w]_\Gamma}$. We prove it by induction on the structure of formula β as follows.
Basis: Suppose that $\beta \in \Gamma$ and β is an atomic formula. we have the following cases:

- If $\beta = p$ is a proposition, then $\models^{\mathcal{M}}_w p$ iff $w \in \Phi(p)$ iff by the definition of $\mathcal{M}_\Gamma$, $[w]_\Gamma \in \Phi_\Gamma(p)$ iff $\models^{\mathcal{M}_\Gamma}_{[w]_\Gamma} p$.

- If $\beta = p(C_1, ..., C_n)$ is an n-ary predicate, then $\models^{\mathcal{M}}_w p(C_1, ..., C_n)$ iff $w \in \Phi(p(C_1, ..., C_n))$ iff by the definition of Φ in semantics, $\prod_{i=1}^{n} [\![C_i]\!]^I_w \subseteq [\![p]\!]^I_w$ iff by the definition of $\mathcal{M}_\Gamma$, $\prod_{i=1}^{n} [\![C_i]\!]^{I_\Gamma}_{[w]_\Gamma} \subseteq [\![p]\!]^{I_\Gamma}_{[w]_\Gamma}$ iff by definition of Φ_Γ in semantics, $[w]_\Gamma \in \Phi_\Gamma(p(C_1, ..., C_n))$ iff $\models^{\mathcal{M}_\Gamma}_{[w]_\Gamma} p(C_1, ..., C_n)$.

- If $\beta = C_1 \sqsubseteq C_2$ is a subsumption relationship, then $[\![C_1]\!]^I_w \subseteq [\![C_2]\!]^I_w$ iff by Lemma 5.3.3, $[\![C_1]\!]^{I_\Gamma}_{[w]_\Gamma} \subseteq [\![C_2]\!]^{I_\Gamma}_{[w]_\Gamma}$ iff $\models^{\mathcal{M}_\Gamma}_{[w]_\Gamma} C_1 \sqsubseteq C_2$.

- If $\beta = C_1 \equiv C_2$ is a definition, then the proof is analogous to case $\beta = C_1 \sqsubseteq C_2$.

- If $\beta = C(a)$ is a concept assertional formula, then $\models^{\mathcal{M}}_w C(a)$ iff $w \in \Phi(C(a))$ iff $[\![a]\!]^I_w \in [\![C]\!]^I_w$ iff by the definition of $\mathcal{M}_\Gamma$, $[\![a]\!]^{I_\Gamma}_{[w]_\Gamma} \in [\![C]\!]^{I_\Gamma}_{[w]_\Gamma}$ iff $\models^{\mathcal{M}_\Gamma}_{[w]_\Gamma} C(a)$.

- If $\beta = R(a,b)$ is a role assertional formula, then the proof is similar to case $\beta = C(a)$.
- If $d_1 \preceq d_2$ is a subdomain relationship. Since $J_\Gamma = J$ and regarding the definition of subdomain relation, we can easily prove that if $\models_w^{\mathcal{M}} d_1 \preceq d_2$, then $\models_{[w]_\Gamma}^{\mathcal{M}_\Gamma} d_1 \preceq d_2$.

Induction Step: We suppose that the theorem holds for formulae $\beta_1, \beta_2 \in \Gamma$.

- If $\beta = \beta_1 \wedge \beta_2$, then $\models_w^{\mathcal{M}} \beta_1 \wedge \beta_2$ iff by proposition 4.2.1, $\models_w^{\mathcal{M}} \beta_1$ and $\models_w^{\mathcal{M}} \beta_2$ iff by the induction hypothesis $\models_{[w]_\Gamma}^{\mathcal{M}_\Gamma} \beta_1$ and $\models_{[w]_\Gamma}^{\mathcal{M}_\Gamma} \beta_2$ iff $\models_{[w]_\Gamma}^{\mathcal{M}_\Gamma} \beta_1 \wedge \beta_2$.
- We have the proof similar to case $\beta = \beta_1 \wedge \beta_2$ for cases $\beta = \beta_1 \vee \beta_2$, $\beta = \beta_1 \rightarrow \beta_2$, and $\beta = \beta_1 \leftrightarrow \beta_2$.
- If $\beta = \neg\beta_1$, then $\models_w^{\mathcal{M}} \neg\beta_1$ iff $\not\models_w^{\mathcal{M}} \beta_1$ iff by the induction assumption $\not\models_{[w]_\Gamma}^{\mathcal{M}_\Gamma} \beta_1$ iff $\models_{[w]_\Gamma}^{\mathcal{M}_\Gamma} \beta$.
- If $\beta = \texttt{OB}_{u@d}\beta_1$, where $u \in \mathcal{U}^*$ we have the following proof.

  [If Part] Since, $\texttt{OB}_{u@d}\beta_1 \in \Gamma$ and $\models_w^{\mathcal{M}} \texttt{OB}_u\beta_1$, by the definition of $\approx_\Gamma$, for all $w_1 \in [w]_\Gamma$, we have $\models_{w_1}^{\mathcal{M}} \texttt{OB}_{u@d}\beta_1$. In other words, for each $z \in J(d)$, $\forall x \in [w]_\Gamma.\forall y.$ if $\langle x, y\rangle \in \Lambda(u)(z)$, then $\models_y^{\mathcal{M}} \texttt{OB}_u\beta_1$.
  Also, for each $z \in J_\Gamma(d)$ and each $l \in W_\Gamma$, if $\langle [w]_\Gamma, l\rangle \in \Lambda_\Gamma(u)(z)$, by the definition of Λ_Γ and Lemma 5.3.4, we have $\exists x, y \in W.\ x \in [w]_\Gamma \wedge y \in l \wedge \langle x, y\rangle \in \Lambda(u)(z)$ and following the previous paragraph, we get $\models_y^{\mathcal{M}} \texttt{OB}_{u@d}\beta_1$, which holds if and only if by the induction hypothesis, $\models_{[y]_\Gamma}^{\mathcal{M}_\Gamma} \beta_1$ that is equals to $\models_l^{\mathcal{M}_\Gamma} \beta_1$.
  Thus, for each $z \in J(d)$, for all $l \in W_\Gamma$, if $\langle [w]_\Gamma, l\rangle \in \Lambda_\Gamma(u)(z)$, then $\models_l^{\mathcal{M}_\Gamma} \beta_1$, and hence, $\models_{[w]_\Gamma}^{\mathcal{M}_\Gamma} \texttt{OB}_{u@d}\beta_1$.

  [Only If Part] For each $z \in J(d)$ and all $w' \in W_\Gamma$, if $\langle w, w'\rangle \in \Lambda(u)(z)$, then by the definition of Λ_Γ, we have $\langle [w]_\Gamma, [w']_\Gamma\rangle \in \Lambda_\Gamma(u)(z)$, and by the assumption that $\models_{[w]_\Gamma}^{\mathcal{M}_\Gamma} \texttt{OB}_{u@d}\beta_1$ (which means for all $l \in W_\Gamma$, if $\langle [w]_\Gamma, l\rangle \in \Lambda_\Gamma(u)(z)$, then $\models_l^{\mathcal{M}_\Gamma} \beta_1$) we have $\models_{[w']_\Gamma}^{\mathcal{M}_\Gamma} \beta_1$, which holds if and only if by the induction hypothesis, we have $\models_{w'}^{\mathcal{M}} \beta_1$.
  Thus, we proved that for each $z \in J(d)$ (regarding to the fact that $J_\Gamma = J$) and all $w' \in W$, if $\langle w, w'\rangle \in \Lambda(u)(z)$, then $\models_{w'}^{\mathcal{M}} \beta_1$, and hence, $\models_w^{\mathcal{M}} \texttt{OB}_{u@d}\beta_1$.
- Since the theorem holds for $\beta = \texttt{OB}_{u@d}\beta_1$, we can easily prove it for cases $\beta = \texttt{PE}_{u@d}\beta_1$, $\beta = \texttt{IM}_{u@d}\beta_1$, and $\beta = \texttt{GR}_{u@d}\beta_1$.

■

Theorem 5.3.6 (The Finite Model Property of MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$) *The MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ logic has the finite model property.*

Proof. Considering the definition of the finite model property (which is mentioned in this section) and Theorem 5.3.5, it is obvious that the MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ logic has the finite model property. ■ In fact MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ has the *strong finite model property.* A logic has the strong finite model property, if it has $F(n)$ size model property, where f is a mapping from integers to integers, and n is the size of the input (here the size of the formula we would like to check its satisfiability) [30]. Since the number of the worlds in the defined finite model $\mathcal{M}_\Gamma$ (w.r.t. $\Gamma = \mathtt{cls}(\alpha)$) is always limited to $O(2^{|\mathtt{cls}(\alpha)|})$, it is obvious that MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ has the strong finite model property. Note that we can easily prove[3] that the size of $\mathtt{cls}(\alpha)$ is of $O(|\alpha|^2)$ and thus, the number of possible worlds in W_Γ in model $\mathcal{M}_\Gamma$ is of $O(2^{|\alpha|^2})$.

Theorem 5.3.7 (Decidability of MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$) *The satisfiability problem for the MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ logic decidable.*

Proof. Since this logic has the strong finite model property, and the set of finite models defined in this section is a recursive set, we can have a general algorithm in which given a formula α in MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$, we generate all the finite models $\mathcal{M}_\Gamma$ with at most $2^{|\alpha|^2}$ possible worlds. Then, in each model, we search for a world where α is true. If we find such a world, the formula is satisfiable, otherwise (α is true in none of the generated worlds) it is not satisfiable.

The proof of this theorem can be presented with another approach. Since this logic has the finite model property (by Theorem 5.3.6) and MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ is axiomatizable, by Theorem 8.5 in [81], MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ is decidable. ■

5.4 Computational Complexity and Expressive Power

There exist a direct relationship between the complexity of the satisfiability problem for a logic and expressive power of its language. In other words, the more expressive power, the

[3]There exist at most $|\alpha|$ number of subformulae of size one, $|\alpha| - 1$ number of subformulae of size two, ..., and one subformula of size $|\alpha|$ in $\mathtt{cls}(\alpha)$. Thus, totally, there are at most $|\alpha|(|\alpha| + 1)/2$ number of subformulae in $\mathtt{cls}(\alpha)$, and hence the size of $\mathtt{cls}(\alpha)$ is of $O(|\alpha|^2)$.

more complexity of the satisfiability problem. Thus, there is a trade-off between the expressive power and algorithmic properties of a logic (such as decidability, and time and space complexity). In the rest of this section, first, we investigate the computational complexity of the MA(DL)^2 logic family, and then, compare its expressive power with other comparable logics.

5.4.1 Computational Complexity

In sections 5.3.1, 5.3.2, and 5.3.3, it is proved that the satisfiability problem for the MA(DL)^2 logic family (with some mentioned constraints which are required for completeness, too) is polynomial time reducible to the satisfiability problem for FO^2. Since, Grädel *et al.* [70] proved that the satisfiability problem for FO^2 is NExpTime-Complete, we can conclude that the upper bound for the complexity of the satisfiability problem for the MA(DL)^2 logic family is NExpTime. Note that this is only an upper bound, and it might be possible to find more efficient algorithms to improve the complexity.

As mentioned in the next section, $\mathcal{ALC}$ is included in MA(DL)^2. Since the satisfiability problem for $\mathcal{ALC}$ is PSpace-Complete, it is clear that the satisfiability problem for the MA(DL)^2 logic family would be in PSpace-Hard. Furthermore, as $\mathcal{ALC}$ is *properly* included in MA(DL)^2, the satisfiability problem for MA(DL)^2 is not PSpace decidable.

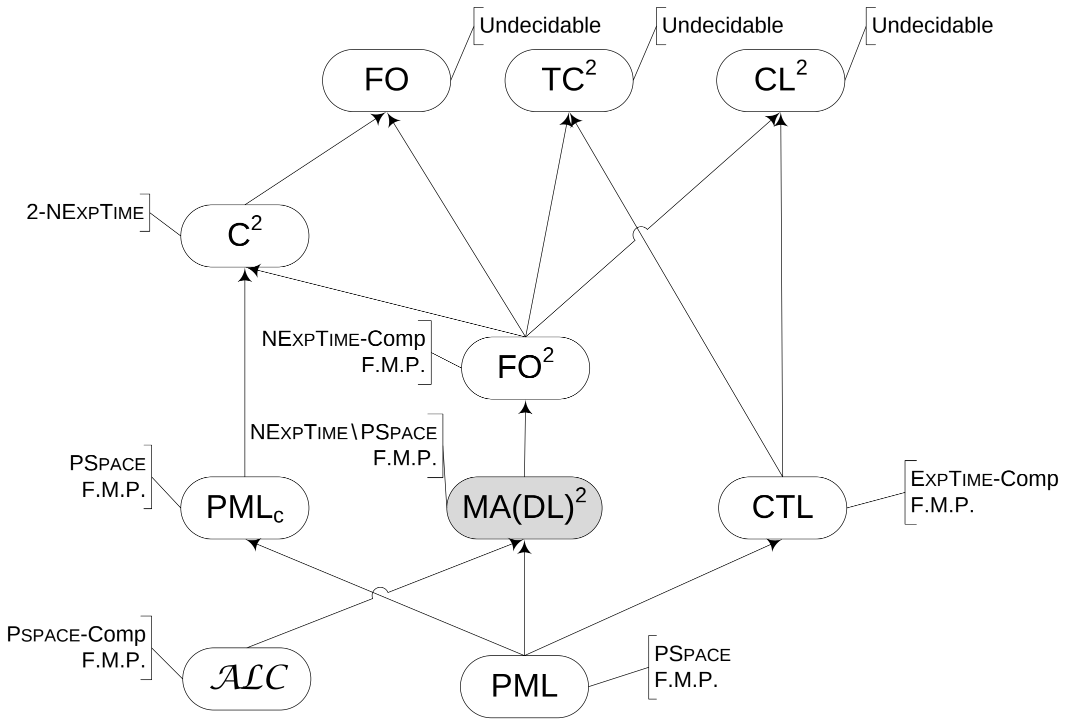
Therefore, the satisfiability problem for MA(DL)^2 is in NExpTime$\setminus$PSpace. Figure 5.1 shows the complexity of the satisfiability problem for the MA(DL)^2 logic family in comparison with the other related logics.

5.4.2 Expressive Power

In the MA(DL)^2 logic family, Core MA(DL)^2 has the least and MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ has the most expressive power as shown in Figure 4.1.

Following the results get in sections 5.3.1, 5.3.2, and 5.3.3, MA(DL)^2 languages are less expressive than FO^2; however, they are more expressive than poly-modal logic (PML) and $\mathcal{ALC}$ description logic. Figure 5.1 shows the expressive power of the MA(DL)^2 logic family in comparison with the other related logic.

To be more precise about the expressive power of MA(DL)^2, the following points are worth mentioning:



F.M.P.= Finite Model Property

$L_1 \rightarrow L_2$= L_2 is more expressive than L_1

FO= First-Order Logic

FO^2= Two-Variable First Order Logic

PML= Poly Modal Logic

PML$_c$= Graded Poly Modal Logic (with graded modalities $\Diamond^{\geq m}$, $\Diamond^{\leq m}$, $\Diamond^{=m}$)

C^2= Two-Variable Counting Logic (with counting quantifiers $\exists^{\geq m}$, $\exists^{\leq m}$, $\exists^{=m}$)

TC^2= FO^2 + Transitive Closures TC$\varphi(x, y)$: there is a φ-path from x to y

CL^2= FO^2 + Reachability $<\varphi(x, y)>\psi(y)$: there is a φ-path from x to y s.t. $\psi(y)$

CTL= Computational Tree Logic

$\mathcal{ALC}$= Attributive Language with Complements

Figure 5.1: Expressive power and properties of the MA(DL)^2 logic family in comparison with the other related logics.

- Since the MA(DL)^2 logic family can be translated into FO^2, but the inverse does not hold, we can conclude that MA(DL)$^2 \subsetneq$ FO^2.

- As mentioned in Section 4.1, the core of the MA(DL)^2 logic family is founded based on the combination of $\mathcal{ALC}$ description logic and multi-authority (poly-modal) version of deontic logic. Therefore, all formulae in poly-modal logic (PML) can be stated in Core MA(DL)^2 as well as the other members of the MA(DL)^2 logic family; however, there are many kinds of formulae (such as concept predicates, terminological and assertional formulae), which cannot be expressed in PML. Thus, we have PML$\subsetneq$ MA(DL)^2. Since terminological and assertional formulae in MA(DL)^2 are defined based on $\mathcal{ALC}$ description logic, it is clear that $\mathcal{ALC} \subsetneq$ MA(DL)^2.

5.5 Summary

Following the discussion on the properties of the MA(DL)^2 logic family in this chapter, the following results are obtained.

- Soundness: the soundness of proof theory of all members of the MA(DL)^2 logic family are proved.

- Completeness: Core MA(DL)^2 and MA(DL)$^2[^{-}_{\mathcal{D}}]$ are proved to be complete, while the other members of the MA(DL)^2 logic family with some constraints are proved to be complete.

- Decidability: two approaches are taken in this chapter to prove the decidability of this logic family; reducing the satisfiability problem for MA(DL)^2 to the satisfiability problem for FO^2, and constructing a finite model using the filtration method. Both of these approaches show the decidability of the MA(DL)^2 logic family.

- Expressive Power: Core MA(DL)^2 has the least and MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ has the most expressive power in this logic family. Also, the MA(DL)^2 logic family is more expressive than $\mathcal{ALC}$ description logic and poly-modal logic (PML and more precisely the KD$_n$ system in this logic). However, it is less expressive than FO^2 logic.

- Computational Complexity: the satisfiability problem for the MA(DL)^2 logic family is in NExpTime $\setminus$ PSpace.

Chapter 6

MA(DL)^2 based Authorization Model

Considering the semantic relationships beside the distribution of semantic-aware environments (SAEs) makes the authorization and access control a complicated problem in them.

Since SAEs are distributed environments that are constructed over a logical foundation (i.e., description logic), and we need to infer implicit policies from the explicit ones based on the semantic relationship defined in the abstract (semantic) layer of SAEs, we inclined to employ a logic for policy specification and inference in the proposed authorization model for SAEs. Using logic for this purpose ensures the soundness and consistency of policy inference in the proposed model as well.

In this chapter the MA(DL)^2-AM authorization model is introduced for authorization in SAEs. The model is founded based on the MA(DL)^2 logic family and it is possible to specify policy rules at the both conceptual and ground (individual) levels in distributed manner. Supporting cooperative security management for shared subdomains, context-awareness, and ability to derive the implicit policy rules from the explicit ones based on the semantic relationships defined in subjects, objects (resources), and actions ontologies are other important characteristics of the proposed model for SAEs in this chapter.

6.1 Authorization Model — Formal Specification

Following the overall framework proposed for security in SAEs, and narrative description of overall models of its main elements, we formally define our authorization model for SAEs, named MA(DL)^2-AM, in the rest of this section.

Definition 6.1.1 (MA(DL)^2 Authorization Model) *MA(DL)^2 Authorization Model for SAEs is a 4-tuple $\langle$FDS, SDS, PAO, ACP $\rangle$, where:*

- *FDS=$\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{DS}, \mathcal{X}, \mathcal{U}, \mathcal{D} \rangle$ is the fundamental data set of the model containing the following elements:*
  - *$\mathcal{S}$ (resp. $\hat{\mathcal{S}}$) as a type (resp. set) of subjects,*
  - *$\mathcal{O}$ (resp. $\hat{\mathcal{O}}$) as a type (resp. set) of objects or resources,*
  - *$\mathcal{A}$ (resp. $\hat{\mathcal{A}}$) as a type (resp. set) of actions or operations on resources,*
  - *$\mathcal{DS}$ as the set of deontic statuses ($\mathcal{DS}$ = {`OB` (obligatory that), `PE` (permissible that), `IM` (impermissible that), `GR` (gratuitous that)}),*
  - *$\mathcal{X}$ as a set of contextual propositions,*
  - *$\mathcal{U}$ as a set of primitive authorities (as mentioned in Section 4.3.1, $\mathcal{U}^*$ denotes the closure set of composite authorities obtained from $\mathcal{U}$),*
  - *$\mathcal{D}$ as a finite set of security domain names.*
- *SDS is a mapping function, which maps each security domain (such as $d_i \in \mathcal{D}$) to the related elements of the domain. The formal specification of security domain is presented in the rest.*
- *PAO={hFPR, dFPR, CPR, WPC, SPC} is a set of policy administration operators which are used by the primitive authorities. In this set, FPR stands for forced policy revision, CPR for consistent policy revision, WPC for weak policy contraction, and SPC for strong policy contraction. More details are described in Section 6.3.2.*
- *ACP is an access control procedure that is used by security agents to infer and enforce security policy rules. The detailed steps of ACP are presented in Section 6.3.3.*

To formally define a security domain, we need to define an MA(DL)^2 signature, which is required for specifying security knowledge base in this model. The signature is defined based on the elements of FDS. Signature $\mathcal{S}ig = \langle \Sigma, \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{I}, \mathcal{U}, \mathcal{D} \rangle$ in this model is defined as follows:

- $\Sigma = \{\mathcal{S}, \mathcal{O}, \mathcal{A}\}$ includes subjects, objects, and actions concept-types.

- $\mathcal{C} = \mathcal{C}_{\mathcal{S}} \cup \mathcal{C}_{\mathcal{O}} \cup \mathcal{C}_{\mathcal{A}}$, where $\mathcal{C}_{\mathcal{S}}$ is a set of subject concepts, $\mathcal{C}_{\mathcal{O}}$ is a set of concepts of accessible objects or resources, $\mathcal{C}_{\mathcal{A}}$ is a set of concepts of possible actions on the objects or resources.
- $\mathcal{R}$ is a set of roles or attributes on concepts $\mathcal{C}$.
- $\mathcal{P}$ includes the set of contextual propositions in FDS, a triple predicate *do*, and a binary predicate *cap*. These predicates are defined in the rest, in Section 6.2.
- $\mathcal{I}$ is the union of the instances of the subjects, objects (resources), and actions, i.e., $\mathcal{I} = \hat{\mathcal{S}} \cup \hat{\mathcal{O}} \cup \hat{\mathcal{A}}$.
- $\mathcal{U}$ is equal to $\mathcal{U}$ in FDS.
- $\mathcal{D}$ is equal to $\mathcal{D}$ in FDS.

Definition 6.1.2 (Security Domain) *Each security domain with the name $d_i \in \mathcal{D}$ is defined with a 4-tuple $\langle u_i, O_i, \mathcal{K}_i, MP_i \rangle$, where*

- *$u_i \in \mathcal{U}^*$ is the authority of the security domain (whose policy rules are applied),*
- *$\mathcal{K}_i$ is the local SKB of the security domain,*
- *$O_i \subseteq \hat{\mathcal{O}}$ is the set of under protection objects registered in the security domain, and*
- *MP_i is the meta policy of the security domain, which is defined in the following.*

Regarding the above definition of security domains, SDS maps each name d_i of a security domain to a 4-tuple, i.e., $SDS(d_i) = \langle u_i, O_i, \mathcal{K}_i, MP_i \rangle$.

Definition 6.1.3 (Meta Policy) *Meta policy is the policy about the security policy rules and it is defined as a 2-tuple $\langle ResSt, DefAcc \rangle$, where $ResSt \in \{PO, NO\}$ is a resolution strategy (NO and PO strategies are defined in Section 6.3.1), and $DefAcc = \{Grant, Deny\}$ is the default access decision, which is used in access control procedure described in Section 6.3.3.*

Note that each authority can state policy rules over each security domain; however, the policy rules of the authority who is determined for the domain are applied by the security agent.

By the above formal definition of security domains, if a security domain d_i (defined as $SDS(d_i) = \langle u_i, O_i, \mathcal{K}_i, MP_i\rangle$) is a *subdomain* of a security domain d_j (defined as $SDS(d_j) = \langle u_j, O_j, \mathcal{K}_j, MP_j\rangle$), or in other words, we have $d_i \preceq d_j$, then $O_i \subseteq O_j$. Analogously, if d_l (defined as $SDS(d_l) = \langle u_l, O_l, \mathcal{K}_l, MP_l\rangle$) is a *shared subdomain* of d_i and d_j, or in other words, $d_l \preceq d_i$ and $d_l \preceq d_j$, then $O_l \subseteq O_i \cap O_j$.

Example 6.1.1 *Running Case Study- Semantic Virtual Organization:*

There are different semantic-aware environments that can be taken as a case study to employ the proposed model for authorization. Examples of them are Semantic Grid (and Virtual Organization), Semantic Peer-to-Peer, Semantic Social Networks, and Semantic Web. Between these cases, semantic Virtual Organization based on the Semantic Grid platform is considered as a case study, because it uses all the features and capabilities of the model proposed based on the MA(DL)^2 logic.

Grid computing environments provide a platform for multiple institutes to share their resources in a wide range. The Grid technology in combination with the Semantic technology constitutes Semantic Grid *[160, 135, 40], where Grid resources and services are described by explicit semantics. The descriptions in the forms of ontologies enables computers and people to easily discover, aggregate, and use the resources and work in cooperation [160, 123].*

Semantic grid is in fact a multi-institutional (multi-domain) environment, where the authorities (e.g, the owner) of different institutional or individual domains need to specify their authorization policies at the both conceptual (semantics) layer and ground (individual) level. Since a grid environment is a combination of domains of resources with variety of users who are unknown to other domains, it is required to identify users based on their attributes specified in some credentials (see e.g., the approach taken in Globus [60], or its extension for semantic grid oriented e-tourism [165]).

Grid has emerged as a platform that enables multiple institutions/ organizations/ groups/ individuals to build a shared space known as Virtual Organization (VO) to achieve a shared goal by collaboration [59]. A VO encompass users and resources supplied by the different partners for achieving the VOs creation goal. Employing semantic technology in a collaborative problem solving in VOs (based on Semantic Grid platform), entails semantic virtual *organizations (SVO) [102]. In this chapter, we define an SVO as a running case study.*

Company A as an IT company wants to outsource the implementation of one of its IT projects to cheap programmers in the world (with more priority to the native ones). For this purpose, A collaborates with company B as a Grid resource provider company to provide the required resources for programmers in an SVO. In this case, d_B is a security domain of B for its shared resources, d_A is a security domain determined by A for the project, and d_X is the shared subdomain of d_A and d_B.

By the above description, the elements of FDS in the model are defined as follows for this case study.

$$\hat{\mathcal{S}} = \textit{set of programmers}$$
$$\hat{\mathcal{O}} = \{proc_1, ..., proc_{10}, mem_1, ..., mem_{20}, stg_{1G}, .., stg_{10G}\}$$
$$\hat{\mathcal{A}} = \{read, write, lock, unlock, exec, wait\}$$
$$\mathcal{X} = \{isWorkTime, is1stOct09To30thDec09, isHoliday\}$$
$$\mathcal{U} = \{adm_A, adm_B\}$$

The set of security domains is defined as $SDS = \{SD_A, SD_B, SD_X\}$, where shared domain SD_X is defined as follows:

$$SD_X = \langle d_X, adm_A|adm_B, \hat{\mathcal{O}}, \mathcal{K}_X, \langle NO, Deny\rangle\rangle$$

SKB $\mathcal{K}_X$ of domain d_X is defined as $\mathcal{K}_X = \langle \mathcal{TB}_X, \mathcal{AB}_X, \mathcal{SB}_X\rangle$, where $\mathcal{TB}_X = Sub \cup Obj \cup Act$. Sub is subjects ontology and is defined as follows:

$$People \equiv Foreigner \sqcup Native$$
$$Programmer \equiv SysProg \sqcup NetProg$$
$$ForeignProg \equiv Programmer \sqcap Foreigner$$
$$NativeProg \equiv Programmer \sqcap Native$$

Obj is objects ontology and is defined as follows:

$$Memory \equiv VolatileMem \sqcup PermanentMem$$
$$EncryptedStorage \sqsubseteq PermanentMem$$
$$Processor \equiv 32bitsProc \sqcup 64bitsProc$$

Act is actions ontology and is defined as follows:

$$ProcAct$$
$$MemAct \sqsubseteq VolatileMemAct$$

In $\mathcal{K}_X$, $\mathcal{AB}$ is initialized with assertions about the resources registered in the domain, and possible actions as follows:

$$VolatileMem(mem_1), ..., VolatileMem(mem_{10}),$$
$$EncryptedStorage(stg_{1G}), PermanentMem(stg_{2G}), ..., PermanentMem(stg_{10G}),$$
$$32bitsProc(proc_1), ..., 32bitsProc(proc_7),$$
$$64bitsProc(proc_8), ..., 64bitsProc(proc_{10}),$$
$$ProcAct(exec), ProcAct(wait),$$
$$MemAct(read), MemAct(write),$$
$$VolatileMemAct(lock), VolatileMemAct(unlock)$$

$\mathcal{SB}_X$ in SKB $\mathcal{K}_X$ is defined later in this chapter, where the security policy rules are defined.

6.2 Security Policy Specification

In the MA(DL)^2-AM authorization model, we use the MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ language from the MA(DL)^2 logic family for policy specification and inference. The logical foundation of this language enables us to infer implicit security policy rules from the explicit ones based on the semantic relationships defined in the ontologies of subjects, objects, and actions.

We describe in the following how to use MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ to specify policy rules at the both conceptual and ground levels.

6.2.1 Conceptual Level Security Policy

A security policy is a set of security policy rules that are defined at the conceptual level as follows.

Definition 6.2.1 (Conceptual Level Policy Rule) *A conceptual level policy rule is a* *formula in MA(DL)^2 with the schema of the form* $\alpha \to \mathtt{ds}_{u@d} do(S, O, A)$*, where*

- α *is a formula specifying the contextual constraint, and it is a logical combination of the contextual propositions* [1],
- $\mathtt{ds} \in \mathcal{DS}$ *is a deontic normative status,*
- $u \in \mathcal{U}^*$ *is an authority,*
- $d \in \mathcal{D}$ *is the name of the security domain, where the policy rule is defined,*
- $do : (\mathcal{S}, \mathcal{O}, \mathcal{A}) \in \mathcal{P}$ *is a ternary predicate that determines a possible access of a subject to a resource (object) for doing an action,*
- $S : \mathcal{S}$ *of type* $\mathcal{S}$ *is a concept of subjects defined in* $\mathcal{TB}$ *of the SKB,*
- $O : \mathcal{O}$ *of type* $\mathcal{O}$ *is a concept of under protection resources (objects) defined in* $\mathcal{TB}$ *of the SKB, and*
- $A : \mathcal{A}$ *of type* $\mathcal{A}$ *is a concept of actions defined in* $\mathcal{TB}$ *of the SKB.*

Example 6.2.1 *Some samples of the conceptual level policy rules for the SVO case study are as follows:*

- *Regarding the contract between the two companies, company B allows access to the shared resources just during the project (1st Oct. 2009 till 30th Dec. 2009). For example, we have the following rule for the processors.*

$$is1stOct09To30thDec09 \to \mathtt{PE}_{adm_B@d_X} do(SysProg, Processor, ProcAct)$$

- *Foreign programmers cannot access the resources in working time of working days* ($\top_{\mathcal{O}}$ *and* $\top_{\mathcal{A}}$ *refer to concept of all objects and concept of all actions respectively).*

$$\neg isHoliday \wedge isWorkTime \to \mathtt{IM}_{adm_A@d_X} do(ForeignProg, \top_{\mathcal{O}}, \top_{\mathcal{A}})$$

[1] α can be defined more complicated; however, we limit it here.

- *System Programmers are allowed to access the 64 bits processors.*

$$\neg isHoliday \wedge isWorkTime \wedge is1stOct09To30thDec09 \rightarrow$$
$$\mathcal{PE}_{adm_A@d_X} do(SysProg, 64bitsProc, ProcAct)$$

6.2.2 Ground Level Security Policy

The ground level policy is taken into consideration, because we may require to specify finer grained policy rules for instances of the resources registered in the domain. Also we may need to have contextual constraints for access at the ground level beside the ones specified at the conceptual level, and there might exist policy rules from legacy systems at the ground level, which should be imported in the new security system founded on our proposed model.

Definition 6.2.2 (Ground Level Policy Rule) *A ground level policy rule for an action a on an under-protection resource (object) o is specified as a 6-tuple* $\langle \pm u, d, cx, Reqs, Caps \rangle$*, where*

- $(+)$ *represents permission and* $(-)$ *represents prohibition.*
- *u, d, and cx are similar to the ones defined at the conceptual level policy rule.*
- $Reqs = \{S_i | (S_i : \mathcal{S}) \in \mathcal{C}\}$ *is a set of subject concepts. It shows credentials which a subject requires to present in order to access the object. In other words, it determines the classes of subjects (in the ontology defined in* $\mathcal{TB}$*) that are allowed to access the resource.*
- $Caps = \{C_i | (C_i : \mathcal{O}) \in \mathcal{C}\}$ *is a set of attributes that the resource supplies when the security policy is satisfied. In other words, it determines the classes of objects (in the ontology defined in* $\mathcal{TB}$*) that the object belongs to, based on its attributes. It is worthwhile to note that in prohibition rules (where the sign is* $-$*), Caps is empty (i.e.,* $Caps = \{\}$*).*

The security capabilities provided by the resource in a ground level policy rule should satisfy the requirements of the requester in order to the requested access be granted. For example, the requester may need the output data of a web service (the requested action) to be encrypted using a specific algorithm. This requires the resource to be an individual of WithEncryptedOuput concept based on its capabilities.

By representing the under-protection resource o with a nominal[2] concept N_o, and the action a with a nominal concept N_a, we can define a ground level policy rule in the MA(DL)^2 language as follows.

Definition 6.2.3 (Ground Level Policy Rule in MA(DL)^2) *A ground level policy rule for permitting an action a on an under-protection object o (i.e., for $\langle +, u, d, cx, Reqs, Caps \rangle$) is specified as an MA(DL)^2 formula with the schema of the form:*

$$cx \rightarrow (\mathtt{PE}_{u@d} do(\bigsqcap_{S_i \in Reqs} S_i, N_o, N_a) \wedge \mathtt{OB}_{u@d} cap(\bigsqcap_{C_i \in Caps} C_i, N_o))$$

, where cap is a binary predicate that assigns the capabilities of a resource to the resource, and a rule for prohibiting the action (i.e., for $\langle -, u, d, cx, Reqs, \{\} \rangle$) is specified as a formula of the form:

$$cx \rightarrow \mathtt{IM}_{u@d} do(\bigsqcap_{S_i \in Reqs} S_i, N_o, N_a)$$

Note that stating the ground level policy rules in MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ has many advantages, such as deriving ground level policy rules for shared resources in shared subdomains using MA(DL)^2 proof theory (for composite authorities), simplifying the access control procedure using the MA(DL)^2 inference service, and uniforming security policy specification at the conceptual and ground levels.

Example 6.2.2 *Some samples of the ground level policy rules for the SVO case study that are specified by company B (which is the real owner of resources) are as follows:*

- *Foreign programmers cannot have write access to the storage with 10 GB capacity.*

$$\mathtt{IM}_{adm_B @ d_X} do(ForeignProg, N_{stg_{10G}}, N_{write})$$

[2]A nominal is a special concept that has only one individual [13]. We assume that the name of a nominal of an individual is a unique name made based on the identity (name) of the individual.

- *Programmers can write their data on the storage with 1 GB capacity encrypted. Encryption of data is the capability of resource stg_{1G}.*

$$1stOct09To30thDec09 \rightarrow \mathtt{PE}_{adm_B@d_X} do(Programmer, N_{stg_{1G}}, N_{write}) \wedge$$
$$\mathtt{OB}_{adm_B@d_X} cap(EncryptedMem, N_{stg_{1G}})$$

6.3 Security Policy Administration and Enforcement

Conflicts between policy rules is an important issue that might be handled to have a sound and reliable security system. This issue has a considerable effect on security policy base administration and also on policy enforcement. Hence, in the rest, at first we have a discussion on the conflict types in MA(DL)^2-AM and our approach on resolving the conflicts. Then we describe the administration operators and access control procedure in the proposed model.

6.3.1 Conflict Types and Resolution

A conflict arises when the objectives of two or more active policy rules can not be met simultaneously. Since in MA(DL)^2-AM, primitive authorities are independent from each other, the conflicts between the policy rules of different authorities do not make any problem. Also note that, in cooperative management, existence of such conflicts does not make any problem in policy inference for composite authorities. Thus, conflicts might occur only in policy space of an authority.

The possible conflicts in policy space of an authority can be classified into

1. the intra-level conflicts, i.e., between the policy rules of the same level (ground or conceptual level), and

2. the inter-level conflicts, i.e., between the policy rules of the ground level with the ones in the conceptual level.

In the rest, we just consider intra-level conflicts and resolve the inter-level ones by taking a special sequence for the steps of the access control procedure. The intra-level conflicts are of two types; *persistent conflicts* and *potential conflicts*. To define such sorts of conflict, we require to define *modal conflicts* in advance.

Definition 6.3.1 *A pair of normative statuses has* modal conflict *if they are completely modally contradictory. The possible conflicting pairs of normative statuses are* $\langle \mathtt{PE}, \mathtt{IM}\rangle$, $\langle \mathtt{OB}, \mathtt{IM}\rangle$, $\langle \mathtt{OB}, \mathtt{GR}\rangle$.

Definition 6.3.2 (Persistent Conflict) *Two policy rules are persistently in conflict, if they are in conflict in all circumstances. Formally, there is a* persistent conflict *between a pair of policy rules* $p_1 = \alpha_1 \rightarrow \mathtt{ds}_{u@d_1} do(S_1, O_1, A_1)$ *and* $p_2 = \alpha_2 \rightarrow \mathtt{ds'}_{u@d_2} do(S_2, O_2, A_2)$*, if*

1. $\vdash \alpha_1 \leftrightarrow \alpha_2$,
2. *there exists* S, O*,* A*, and* d *such that* $\mathcal{TB} \cup \{\mathtt{ds}_{u@d_1} do(S_1, O_1, A_1)\} \vdash \mathtt{ds}_{u@d} do(S, O, A)$ *and* $\mathcal{TB} \cup \{\mathtt{ds'}_{u@d_2} do(S_2, O_2, A_2)\} \vdash \mathtt{ds'}_{u@d} do(S, O, A)$,
3. $\mathtt{ds}$ *and* $\mathtt{ds'}$ *are one of the possible conflicting pairs of normative statuses.*

For example, policy rules $\neg isWorkTime \rightarrow \mathtt{PE}_{adm_A@d_X} do(Programmer, Memory, MemAct)$ and $\neg isWorkTime \rightarrow \mathtt{IM}_{adm_A@d_X} do(NetProg, VolatileMem, MemAct)$ are persistently in conflict.

Definition 6.3.3 (Potential Conflict) *There is a potential conflict between two policy rules, if they are in conflict in some contextual conditions. Formally, there is a* potential conflict *between a pair of policy rules* $p_1 = \alpha_1 \rightarrow \mathtt{ds}_{u@d_1} do(S_1, O_1, A_1)$ *and* $p_2 = \alpha_2 \rightarrow \mathtt{ds'}_{u@d_2} do(S_2, O_2, A_2)$*, if* $\vdash \alpha_1 \not\leftrightarrow \alpha_2$*,* $\alpha_1 \neq \neg\alpha_2$*, and conditions (2) and (3) in Definition 6.3.2 are fulfilled.*

For example two policy rules $\neg isWorkTime \rightarrow \mathtt{PE}_{adm_A@d_X} do(Programmer, Memory, MemAct)$ and $\neg isHoliday \rightarrow \mathtt{IM}_{adm_A@d_X} do(NetProg, VolatileMem, MemAct)$ are not always in conflict. However, when it is a working time in a working day, both policy rules are activated and a conflict occurs.

The time points where the aforementioned conflicts might occur are as follows.

- A persistent conflict might occur in adding a new policy rule to an SKB. More formally, it occurs when we want to add a new policy rule $p_{new} = \alpha \rightarrow \mathtt{ds}_{u@d}(S, O, A)$ to an SKB $\mathcal{K}$, and we have $\mathcal{K} \vdash \neg p_{new}$. In other words $\mathcal{K} \cup \{p_{new}\} \vdash \mathtt{F}$, which means p_{new} has conflict with some of the existing policy rules.

Table 6.1: Conflict resolution based on the NO and PO strategies.

Conflicting Pair	**Overriding Relation Label**	NO **Strategy**	PO **Strategy**
$\langle \texttt{PE}, \texttt{IM} \rangle$	$\texttt{PE} \xrightarrow{PO} \texttt{IM}, \quad \texttt{PE} \xleftarrow{NO} \texttt{IM}$	IM	PE
$\langle \texttt{OB}, \texttt{IM} \rangle$	$\texttt{OB} \xrightarrow{PO} \texttt{IM}, \quad \texttt{OB} \xleftarrow{NO} \texttt{IM}$	IM	OB
$\langle \texttt{OB}, \texttt{GR} \rangle$	$\texttt{OB} \xrightarrow{PO} \texttt{GR}, \quad \texttt{OB} \xleftarrow{NO} \texttt{GR}$	GR	OB

- A conflict might occur in access control time (by receiving an access request) and updating a contextual information. In this step, some potential conflicts might be realized. More precisely, if we update an SKB $\mathcal{K}$ with the new contextual facts, the SKB becomes inconsistent, and thus $\mathcal{K} \vdash \texttt{F}$.

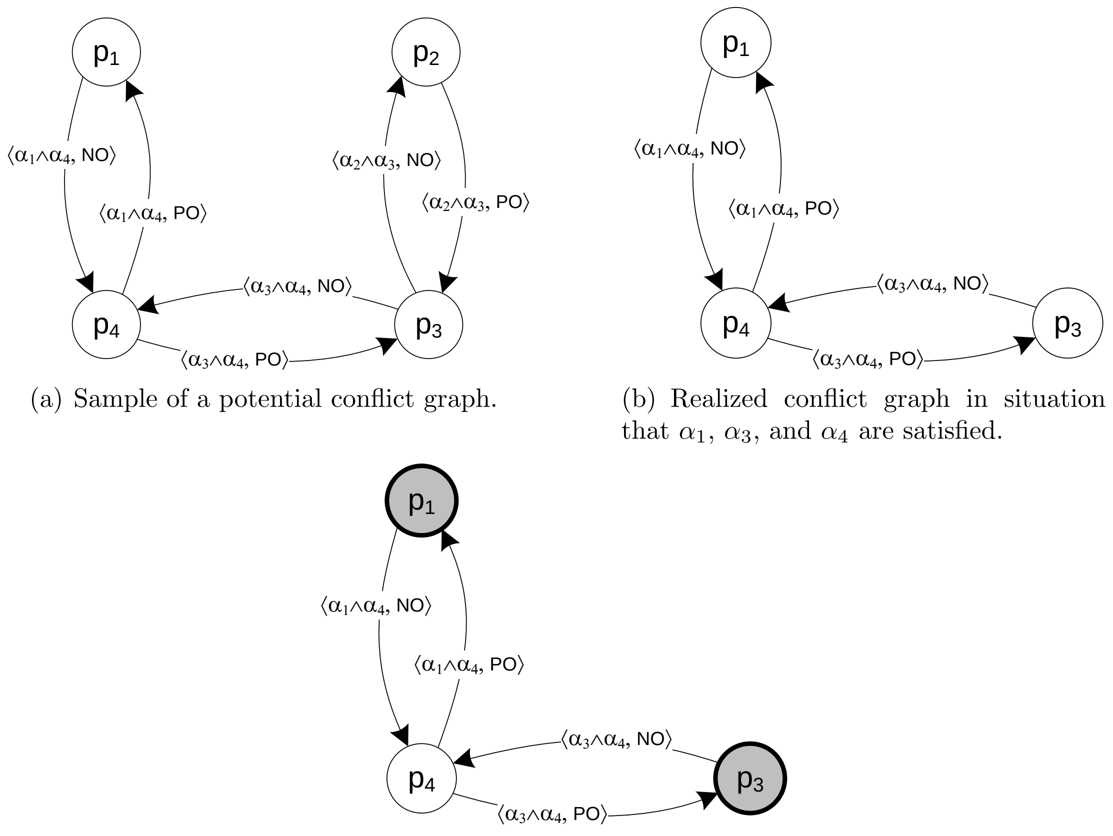
We can easily prevent the occurrences of persistent conflicts in the operators used for adding a new policy (see Section 6.3.2). However, detecting and resolving potential conflicts should be performed dynamically in access control procedure. For this purpose, we construct a potential conflict graph (similar to the approach we proposed in [114]) when adding policy rules to the SKB. We use this graph to detect the realized conflicts and resolve them.

To resolve the conflicts using the potential and realized conflict graphs, we should first determine the possible conflict resolution strategies. Analyzing the modal conflicting pairs in Definition 6.3.1 shows that they are originated from one of the $\langle \texttt{OB}\neg p, \texttt{OB}\neg p \rangle$ or $\langle \texttt{OB}\neg p, \neg\texttt{OB}\neg p \rangle$ primitive conflicting pairs. On this basis, we can define two strategies for conflict resolution including:

- **Negative Obligation takes precedence (NO) strategy:** This is a pessimistic strategy, which aims at decreasing access to the under-protected objects to increase the security assurance. In this strategy, $\texttt{OB}\neg p$ and $\neg\texttt{OB}p$ override $\texttt{OB}p$ in the primitive conflicting pairs. The results of following such a strategy is shown in Table 6.1.

- **Positive Obligation takes precedence (PO):** Unlike the NO strategy, the PO strategy is an optimistic strategy. It aims at providing maximum accessibility to the resources and executing actions. In this strategy, $\texttt{OB}p$ overrides $\texttt{OB}\neg p$ and $\neg\texttt{OB}p$ in the primitive conflicting pairs. The results of following such a strategy is shown in Table 6.1.

As mentioned earlier, we use two following graphs for detecting and resolving the conflicts in access control procedure of the MA(DL)^2-AM authorization model.



(a) Sample of a potential conflict graph.

(b) Realized conflict graph in situation that α_1, α_3, and α_4 are satisfied.

(c) Conflict resolution based on PO strategy. Bolded vertices should be removed.

Figure 6.1: Potential conflict detection and resolution steps.

Definition 6.3.4 (Potential Conflict Graph) *A potential conflict graph is an edge-labeled directed graph such that each vertex represents a policy rule that is potentially in conflict with another policy rule, and each edge represents a potential conflict between a pair of policy rules* $p_1 = \alpha_1 \rightarrow \mathtt{ds}_{u@d_1} do(S_1, O_1, A_1)$ *and* $p_2 = \alpha_2 \rightarrow \mathtt{ds'}_{u@d_2} do(S_2, O_2, A_2)$*, and has a label of the form* $\langle \alpha_1 \wedge \alpha_2, OR \rangle$*.* $\alpha_1 \wedge \alpha_2$ *determines the contextual condition in which the potential conflict becomes realized, and* OR *determines overriding relation and is one of* NO *(negative obligation precedence) and* PO *(positive obligation precedence).* OR *is determined based on the second column of Table 6.1.* NO *and* PO *are used for conflict resolution and are defined for normative statuses (belong to* $\mathcal{DS}$*) in Table 6.1.*

Figure 6.1(a) shows a sample of potential conflict graph.

Definition 6.3.5 (Realized Conflict Graph) *A realized conflict graph is a subgraph of a potential conflict graph. It contains only the edges that the first element of their labels are satisfied based on the current contextual information in the SKB.*

Figure 6.1(b) shows the realized conflict graph of the graph shown in Figure 6.1(a) in a special contextual condition.

For dynamic resolution of realized conflicts, we just need to remove the policy rules on the tails of the edges that the second elements of their labels are equal to the determined conflict resolution strategy in the meta policy of the security domain (i.e., in $ResSt$ in meta policy MP of the security domain). For example, if we have the PO strategy, the filled vertices (policy rules) in Figure 6.1(c) are removed temporarily from the SKB for access decision.

6.3.2 Security Policy Base Administration

In MA(DL)^2-AM, each authority can administrate his specified security policy rules in his/her security domain using the following administration operators provided through the PAP's interface. Note that we can use the following operators for the policy rules of each level (i.e., conceptual and ground) separately.

Policy Revision Operators: Add policy rules to the SKB, and update the potential conflict graph if it is required. These operators are as follows.

- FPR (Forced Policy Revision): Adds a new policy rule by removing all the conflicting old policy rules. We have two kinds of FPR operator:
  - hFPR: harmonizes the conflicting policy rules w.r.t. the new one. Harmonizing a policy rule $p_1 = \alpha \rightarrow \mathtt{ds}_{u@d_1} do(S_1, O_1, A_1)$ regarding the new policy rule $p_{new} = \alpha \rightarrow \mathtt{ds}_{u@d_2} do(S_2, O_2, A_2)$ results in a policy rule $p_1' = \alpha \rightarrow \mathtt{ds}_{u@d_1} do(S_1 \sqcap \neg S_2, O_1 \sqcap \neg O_2, A_1 \sqcap \neg A_2)$, which replaces p_1.
  - dFPR: deletes or removes the policy rules conflicting with the new one.
- CPR (Consistent Policy Revision): Adds a new policy rule only if it is consistent with the existing ones in the SKB.

Policy Contraction Operators: Remove policy rules from the SKB, and update the potential conflict graph if it is required.

- WPC (Weak Policy Contraction): Removes an explicit policy rule existing in the SKB.
- SPC (Strong Policy Contraction): Removes a policy rule and all the policy rules entails such a rule. Note that the determined policy rule might be an explicit rule existing in the SKB, or might be an implicit rule, which is inferred from the other ones.

6.3.3 Access Control Procedure

The access control procedure, which is used by the security agent of a security domain, has the following steps. Suppose that the following procedure is employed in a security domain d, such that $SDS(d) = \langle u, O, \mathcal{K}, MP \rangle$.

1. **Request Reception:** PEP receives an access request in the form $\langle s_r, o_r, a_r, Crd_r, Req_r \rangle$ from a subject, where s_r, o_r, and a_r are subject (access requester), resource (object), and requested action respectively. Crd_r is the set of credentials that s_r presents, and Req_r is the set of requirements that s_r requires to be satisfied by the capabilities of the requested resource.

2. **Request Validation:**

   (a) PEP checks whether o_r is registered in the security domain, and a_r is an eligible action on o_r (by checking the description of the resource's Web service specified in OWL-S).

   (b) PEP checks the validaty of the received credentials using Credential Verifier and Source of Authority (SOA), and sends the validated credentials to PDP.

3. **SKB Initialization:**

   (a) PDP generates a set of assertions based on the received credentials and inserts them to $\mathcal{AB}$ of SKB $\mathcal{K}$. The assertions are of the form $C(s_r)$, where $C \in Crd_r$. For example if it receives a credential that says "Ali is a student" and concept "Student" is defined in $\mathcal{TB}$ of SKB $\mathcal{K}$, we add the assertion "Student(Ali)" to $\mathcal{AB}$.

   (b) By the request of PDP, SKB $\mathcal{K}$ updates itself with the last changes in contextual information (using the Context Handler component).

(c) PDP derives the realized conflict graph (from the potential conflict graph constructed during the specification of the security policy rules), performs conflict resolution based on the determined resolution strategy, and removes the selected conflicting policy rules from the SKB.

4. **Ground Level Access Decision Making:** PDP makes initial access decision based on the ground level policy rules of the requested resource in the following steps. N_{o_r} and N_{a_r} are nominals for o_r and a_r respectively.

   (a) If $\mathcal{K} \vdash \texttt{PE}_{u@d}do(\bigsqcap_{crd_i \in Crd_r} crd_i, N_{o_r}, N_{a_r}) \wedge \texttt{OB}_{u@d}cap(\bigsqcap_{req_i \in Req_r} req_i, N_{o_r})$, it means all the requirements of the requester are satisfied by the capabilities of the resource. Therefore, the requester is permissible to receive the requested access based on the ground level policy rules.

   (b) If $\mathcal{K} \vdash \texttt{IM}_{u@d}do(\bigsqcap_{crd_i \in Crd_r} crd_i, N_{o_r}, N_{a_r})$, the access is denied and the access control procedure terminates after executing the last step (SKB Purging).

   (c) If $\mathcal{K} \vdash \texttt{PE}_{u@d}do(\bigsqcap_{crd_i \in Crd_r} crd_i, N_{o_r}, N_{a_r}) \wedge \neg\texttt{OB}_{u@d}cap(\bigsqcap_{req_i \in Req_r} req_i, N_{o_r})$, it means the requester is permissible to take the requested access; however, all of its requirements can not be satisfied. In this case, the security agent can negotiate with the requester and then

   - the requester can cancel the request, and the procedure terminates after executing the last step (SKB Purging), or
   - it can enquire for the available capabilities and then make its final decision (to change the requirements in the request or cancel the request).

5. **Conceptual Level Access Decision Making:** PDP makes its final access decision based on the conceptual level policy rules in the following steps:

   (a) PDP maps s_r, o_r, and a_r to appropriate concepts in the ontologies by the following equations. Note that these concepts are different from the nominals for the individuals. PDP leverages its inference engine over the SKB to infer the most specific concepts ($\mathtt{msc}$[3]) of each of the individuals s_r, o_r, and a_r for this purpose.

$$C_{s_r} = \bigsqcap_{S_i \in \mathtt{msc}(s_r)} S_i \qquad C_{o_r} = \bigsqcap_{O_i \in \mathtt{msc}(o_r)} O_i \qquad C_{a_r} = \bigsqcap_{A_i \in \mathtt{msc}(a_r)} A_i$$

(b) If $\mathcal{K} \vdash \mathtt{PE}_{u@d} do(C_{s_r}, C_{o_r}, C_{a_r})$, the request is granted.

(c) If $\mathcal{K} \vdash \mathtt{IM}_{u@d} do(C_{s_r}, C_{o_r}, C_{a_r})$, the request is denied.

(d) If none of the above inferences are made and default access policy (specified in $DefAcc$ of meta policy MP) is *Deny*, the request is denied; otherwise it is granted.

6. **SKB Purging:** After completing the decision making process, PDP

    (a) removes the current contextual facts from the SKB,

    (b) removes the assertions related to the requester subject from $\mathcal{AB}$ of the SKB, and

    (c) returns back the policy rules that are removed for realized potential conflicts to the SKB.

In the above procedure, the inter-level conflicts do not make any problem, because the policy rules of the two levels are considered in two separated steps. In fact, *negative takes precedence* approach is employed for inter-level conflict resolution.

Example 6.3.1 *As an example, suppose the security agent of domain d_X receives an access request $\langle Ali, stg_{1G}, write, \{SysProg, Native\}, \{EncryptedStorage\}\rangle$ in 10 am of 6th Oct. 2009. It can infer the following formula from the ground level policy rules in the SKB $\mathcal{K}_X$*

$$\mathit{PE}_{(adm_A|adm_B)@d_X} do(SysProg \sqcap Native, N_{stg_{10G}}, N_{write}) \wedge$$
$$\mathit{OB}_{(adm_A|adm_B)@d_X} cap(EncryptedStorage, N_{write})$$

and can infer the following formula from the conceptual level policy rules

$$\mathit{PE}_{(adm_A|adm_B)@d_X} do(SysProg \sqcap Native, PermanentMem, MemAct).$$

Hence, the access request is granted.

[3]Formally, $\mathtt{msc}$ of an individual a of type σ is a set of incomparable concepts $C_1, ..., C_n$ of type σ satisfying $\mathcal{AB} \vdash C_i(a)$ $(1 \leq i \leq n)$ and if there exist a concept D such that $\mathcal{AB} \vdash D(a)$, then there exist a C_i in the above set such that $\mathcal{TB} \vdash C_i \sqsubseteq D$.

Table 6.2: Evaluation of MA(DL)^2-AM in comparison with some other authorization models.

Model	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**
FGAC [46]	G	-	C	Yes	Yes	Yes	$A^{+/-}$	No	IB	G, E	No
XACML [121]	G	-	D	Yes	Yes	Yes	$A^{+/-}$, $O^{+/-}$	No	AB	R, G	No
CLAC [132]	C	O	C	Yes	No	Yes	$A^{+/-}$	No	AB	R, G	No
SBAC [91]	C, G	S, O, A	C	Yes	No	Yes	$A^{+/-}$	No	AB	R, G, E	No
SAC (SPL) [162]	G	-	D	No	Yes	Yes	A^{+}	No	IB	R, G	Yes
Rei [96]	G	-	D	Yes	Yes	Yes	$A^{+/-}$, $O^{+/-}$	Yes	AB	R, G, D	Yes
KAoS [154]	G	A	D	Yes	Yes	Yes	$A^{+/-}$, $O^{+/-}$	Yes	AB	R, G	Yes
MA(DL)^2-AM [7]	C, G	S, O, A	D	Yes	Yes	Yes	$A^{+/-}$, $O^{+/-}$	Yes	AB	R, G, D	Yes

R1=Policy specification granularity level [C: Conceptual level, G: Ground (individual) level]
R2=Policy propagation and inference domains [S: subjects domain, O: objects domain, A: actions domain]
R3=Policy specification and management [D: distributed, C: centralized]
R4=Existence an approach for conflict detection and resolution [Yes, No]
R5=Support of contextual constraints in policy specification [Yes, No]
R6=Is abstract enough to be independent from implementation [Yes, No]
R7=Policy types supported in the model
[$A^{+/-}$: positive/negative authorization, $O^{+/-}$: positive/negative obligation]
R8=Existence of formal semantics for the policy language of the model [Yes, No]
R9=Subject (user) identification in policy rule specification [IB: identity-based, AB: attribute-based]
R10=Administration Facilities [E: exception policy, R: role, G: group, D: delegation]
R11=Has ability for policy composition [Yes, No]

6.3.4 Evaluation of the MA(DL)^2-AM Authorization Model

It is not easy to evaluate a proposed authorization model or access control enforcement mechanism, because there is not a set of acceptable criteria for this purpose. In fact, comparing the different models might not be acceptable, due to the difference in the meaning or definition of security for which a model is proposed. Thus, to the best of our knowledge, the best way to compare the authorization models proposed for the same environment is to determine the authorization requirement for that environment in advance. Such requirements can be used as criteria for comparing the proposed models for such environments. Since we investigated the authorization requirements for SAEs in Section 3.2, the MA(DL)^2-AM authorization model in comparison with other famous models proposed for SAEs, based on these requirements are illustrated in Table 6.2.

The following points in evaluating the proposed model and filling Table 6.2 are taken into account:

(R1, R2, R4, R5, R6, R11) Regarding the specification of the proposed model, it is clear that how the model satisfies the requirements R1 (policy specification in both conceptual and ground levels), R2 (supporting policy propagation and inference based on the ontologies of subjects, objects, and actions), R4 (conflicting rules detection and resolution), R5 (supporting contextual conditions in policy rule specification), R6 (has an acceptable abstraction level), and R11 (policy composition for shared subdomains).

(R3) To satisfy authorization requirement R2 (distributivity of policy specification), the model follows the multi-security-domain structure, where the environment is divided into some security domains and policy specification done in the distributed manner. In each security domain, we may have some authorities who are authorized to specify the security policies, and we can use the cooperative security management approach to policy inference over the distributed policy rules. Note that conceptual level policy rules of each domain is maintained centrally in the SKB of the domain. However, ground-level policy rules of each resource is stored in service provider of the resource (as it is mentioned in canonical model of resources in Section 3.4.3).

(R7) Supporting different deontic (normative) statuses (defined in set $\mathcal{DS}$) in the MA(DL)^2 language for policy rule specification enables the model to support the specification of positive and negative authorization as well as positive and negative obligations.

(R8) The model is based on the MA(DL)^2 logic which is placed in logic layer in the semantic layer cake and it has a clear formal semantics as presented in Chapter 4.

(R9) Each concept in the subjects ontology is a representative of a set of principals (subjects) that have some common attributes. Thus, possibility of the model in specification of policy rules over the subject concepts categorizes it in the class of the attribute-based (credential-based) authorization models. Refer to the description of stage 3(a) in the access control procedure to see how the provided attributes of the requester (by some credentials) used for access decision making.

(R10) Each Role and group in our proposed model can be defined as a concept in subjects ontology. Thus, roles and groups can be defined easily in our model. Note that the hierarchy of roles can be defined, too. If a role R_1 (e.g., Manager) is a subrole of R_2 (e.g., Employee), in our model corresponding concept R_2 should be defined as a subsumer of corresponding concept R_1 in subjects ontology (i.e., $R_1 \sqsubseteq R_2$). It is clear that each subject or user of R_1 is a subject or user of R_2 as well (by relationship $R_1 \sqsubseteq R_2$ and axiom DLA), and each permission assigned to R_2 is assigned to R_1 as well. By delegative cooperative management (provided by MA(DL)$^2[^{\mathcal{U}}_{-}]$ in our model, the administrative delegation is also possible.

6.4 Summary

The main objective of this thesis is presenting an appropriate authorization model for SAEs. In this chapter an authorization model (named MA(DL)^2-AM) based on the MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]$ logic is formally defined. The fundamental elements of the model, as well as the knowledge base administration operators, and access control procedure (which uses the inference service over the security knowledge base) are defined precisely. Since occurring conflicts between the policies specified by different authorities is inevitable, the possible types of conflicts are investigated and prevented or resolved in the proposed model. Finally, MA(DL)^2-AM is evaluated qualitatively in comparison with the analogous models using the criteria extracted from the investigated authorization requirements for SAEs.

Quantitative evaluation of the MA(DL)^2-AM authorization model, using an access control system (of a security agent) implemented based on the access control procedure proposed in this model is presented in the next chapter.

Chapter 7

Implementation and Experimental Results

To show the applicability of the MA(DL)^2-AM authorization model, and also to evaluate it in practice, a simple version of a security agent based on the proposed model has been implemented as a prototype. For this purpose, considerable efforts devoted to the development of the MA(DL)^2 inference engine, where analytic tableaux approach is leveraged. The implemented prototype of the security agent uses this inference engine to do the inferences required in the access control procedure of MA(DL)^2-Am authorization model. The experimental results of evaluating the implemented inference engine and security agent prototype are also presented in this chapter.

7.1 MA(DL)^2 Inference Engine

There are variants of reasoning algorithms which in them analytic tableaux is more match with non-classical logics like MA(DL)^2 [64]. Analytic tableaux is essentially based on the semantics of logical connectors and modal operators in these logics. In this research we used refutation approach in analytic tableaux method. In refutation tableaux reasoning, the objective is to demonstrate that the negation of a given formula cannot be satisfied. In refutation tableaux, we have a tableaux tree with a given negation formula F at its root, and sub-formulae of F at each node. Constructing the tableaux tree is performed based on the tableaux expansion rules, which are introduced for MA(DL)^2 logic in this section. If in all leaves of tableaux tree we reach the contradiction, the tree is closed and the negation formula is not satisfiable.

There are many implementations of theorem provers based on the tableaux method for modal logics [17, 36, 63, 125], and modalized description logics [112]; however, non of them could be employed for reasoning in MA(DL)^2 logic. Some of the well-known theorem provers for modal and deontic logics are MLTP [105], KED [12], KEM [68, 127], and GQML-Prover [151]. The most popular reasoners for description logics are RACER [75], FaCT++ [153], and Pellet [146].

7.1.1 Analytic Tableaux for MA(DL)^2

We use refutation tableaux approach for MA(DL)^2 logic family. In this approach, for checking the validity of a formula α in an SKB $\mathcal{K} = \langle \mathcal{TB}, \mathcal{AB}, \mathcal{SB} \rangle$, we add the negation of α to the SKB and check the consistency of the resulted SKB. Since the tableaux rules presented in this chapter are provided for the inference of c-formulae, it is sufficient to check the unsatisfiability of $\mathcal{SB} \cup \{\neg\alpha\}$ w.r.t. terminological box $\mathcal{TB}$. Note that assertional box $\mathcal{AB}$ is not required for the inference of c-formulae.

We follow set-labeling tableaux approach [76] in our proposed approach in which each node in the tableaux tree is labeled with a set of formulae. The expansion rules determine how to add new nodes to the *leaves* of the tree based on their set-label. In this approach we do not need to consider the labels of nodes in a branch to attach a new node to a leaf, and just the label of the leaf is enough. In our proposed tableaux system, each set-label of a node is a *block*, which is in fact a set of sets. Before introducing the content of the block and also the tableaux expansion rules, we require to introduce some notations.

- X and Y are sets of formulae.
- p, q, l, and dl are formulae
- $\mathtt{OB}X = \{\mathtt{OB}_{u@d}p \mid p \in X, u \in \mathcal{U}^*, d \in \mathcal{D}\}$
- $\mathtt{PE}X = \{\mathtt{PE}_{u@d}p \mid p \in X, u \in \mathcal{U}^*, d \in \mathcal{D}\}$
- $\mathtt{OB}_{u@d}X = \{\mathtt{OB}_{u@d}p \mid p \in X\}$
- $\mathtt{PE}_{u@d}X = \{\mathtt{PE}_{u@d}p \mid p \in X\}$

- $X, p = X \cup p$
- $[TB; F; L; OBF; PEF]$ is a set-block-label, where TB is a set of t-formula of the terminological box of an SKB (including subsumptions and definitions), F is a set of raw formulae, L is a set of non-modal literals, OBF is a set of obligation formulae (in the form $\mathtt{OB}_{u@d}p$), and PEF is a set of permissible formulae (in the form $\mathtt{PE}_{u@d}p$).

$[TB; F; \{\}; \{\}; \{\}]$ is an initial block which is a label of the root of a tableaux tree. In this block, F is a formula which should be proven that it is unsatisfiable; more precisely it is defined as $F = \bigwedge_{p_i \in \mathcal{SB}} p_i \wedge \neg\alpha$, where α is the c-formula in MA(DL)^2 that we like to check its validity. In our approach, F must be in negation normal form (NNF).

Definition 7.1.1 (Negation Normal Form (NNF)) *A formula F is in NNF, if*

- *it only contains logical connectors $\wedge$, $\vee$, and $\neg$, deontic statuses* ***OB*** *and* ***PE****,*
- *negations appear just before atomic formulae, and*
- *arguments of predicates (defined on $\mathcal{ALC}$ concepts) would be in expanded and negation normal form as defined in description logic.*

To convert a c-formula in MA(DL)^2 logic family to a formula in NNF, we use the NNF function. In the following, the definition of NNF for the special formulae in MA(DL)^2 are presented. The NNF function for other formulae can be find in most of the logic text books[1].

$$NNF(\mathtt{OB}_{u@d}F) = NNF_M(\mathtt{OB}_{u@d}(CNF(NNF(F))))$$
$$NNF(\mathtt{PE}_{u@d}F) = NNF_M(\mathtt{PE}_{u@d}(DNF(NNF(F))))$$
$$NNF_M(\mathtt{OB}_{u@d}(F_1 \wedge F_2)) = NNF_M(\mathtt{OB}_{u@d}F_1) \wedge NNF_M(\mathtt{OB}_{u@d}F_2)$$
$$NNF_M(\mathtt{OB}_{u@d}l) = \mathtt{OB}_{u@d}l \text{ (l is a literal)}$$
$$NNF_M(\mathtt{PE}_{u@d}(F_1 \vee F_2)) = NNF_M(\mathtt{PE}_{u@d}F_1) \vee NNF_M(\mathtt{PE}_{u@d}F_2)$$
$$NNF_M(\mathtt{PE}_{u@d}l) = \mathtt{PE}_{u@d}l \text{ (l is a literal)}$$
$$NNF(p(C_1, ..., C_n)) = p(CNF_C(NNF_C(EXP_C(C_1))), ..., CNF_C(NNF_C(EXP_C(C_n))))$$

[1]The full definition of NNF can be found in Prolog code of the MA(DL)^2 reasoner in Appendix B.

In the above definition, two conversion functions CNF and DNF are convertors to conjunction normal form and disjunction normal form, respectively. NNF_M just converts the modal formulae $\mathtt{OB}_{u@d}F$ and $\mathtt{PE}_{u@d}F'$, only if F is in CNF and NNF forms, and F' is in DNF and NNF forms.

In converting predicates p, functions CNF_C and NNF_C convert concepts like C_i (which are the arguments of predicate p) to conjunction and negation normal form in $\mathcal{ALC}$ logic. Function EXP_C expands the description of concepts like C_i based on the definitions exist in $\mathcal{TB}$.

Note that such conversions are necessary to correctly apply the tableaux expansion rules, which are introduced in the rest.

The tableaux expansion rules for MA(DL)^2 logic family, are written in the way that we have the following steps by running the proposed tableaux method. Each step is resulted from applying some of the proposed rules.

Step 1 *Fragmentation:* fragmenting the formulae to their subformulae by applying the α and β rules.

Step 2 *Categorization:* categorizing atomic sub-formulae (resulted from the complex ones after fragmentation) in their categories (i.e. sets L, OBF, and PEF) by applying the γ rules.

Step 3 *Sub-Tableaux Construction:* existing modal formulae results in constructing sub-tableaux trees by applying the κ and θ rules. In fact, each sub-tableaux tree run the tableaux method in a new possible world, and the output resulted from running the method in the new world affects the output of its running in the world accessing to the new world. The description of the rules in the rest shows that the κ rule can never be applied after the θ rule in a tableaux tree.

Step 4 *Closure:* closes the branches of the tableaux tree when some conditions are satisfied. Applying the did_1, did_2, did_3, iid_1, and iid_2 rules close the branches in the tableaux tree.

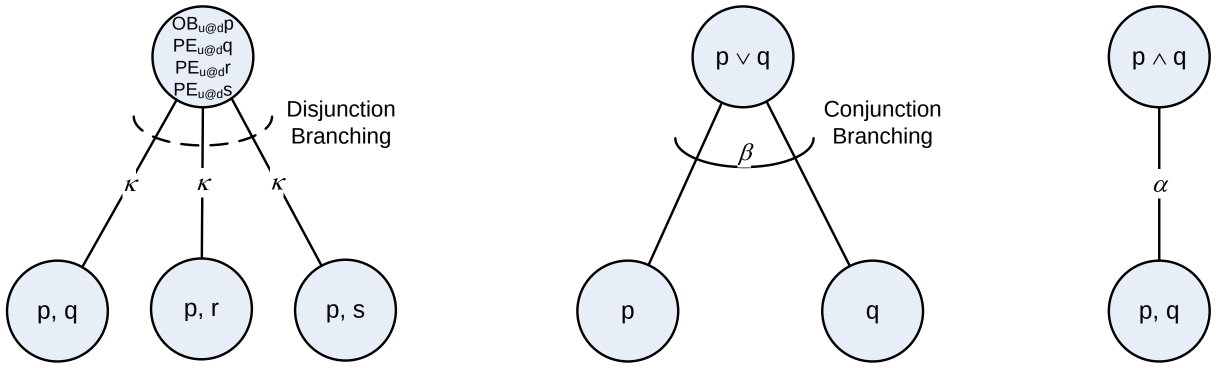


Figure 7.1: The impact of applying the α, β, and κ rules on constructing an MA(DL)^2 tableaux tree.

7.1.2 Tableaux Expansion Rules for MA(DL)^2

The expansion rules, which are employed in our proposed tableaux approach to break down a tableaux tree, are as follows.

Fragmentation Rules

Rules α and β are used to decompose conjunction and disjunction formulae respectively in F. In rule α, a new node for tableaux tree is created; however, in rule β, two new nodes are created. In fact, applying the β rule, cause a conjunction branching in the tree, since for closing the node, both subtrees should be closed. Figure 7.1 shows the effect of applying these two rules graphically.

$$\text{(Rule } \alpha) \frac{[TB; F, p \wedge q; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F, p, q; L; \texttt{OB}\ X; \texttt{PE}\ Y]}$$

$$\text{(Rule } \beta) \frac{[TB; F, p \vee q; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F, p; L; \texttt{OB}\ X; \texttt{PE}\ Y] \mid [TB; F, q; L; \texttt{OB}\ X; \texttt{PE}\ Y]}$$

Rules ξ_{JO} and ξ_{JP} in combination with the α and β rules, decompose the modal formulae containing disjunctive composite authorities (in the form of $\texttt{ds}_{(u_1|u_2)@d}\ p$). Since in NNF, we just have OB and PE statuses, we have the fragmentation rules for these statuses as follows.

$$\text{(Rule } \xi_{JO}) \frac{[TB; F, \texttt{OB}_{(u_1|u_2)@d}p; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F, \texttt{OB}_{u_1@d}p \wedge \texttt{OB}_{u_2@d}q; L; \texttt{OB}\ X; \texttt{PE}\ Y]}$$
$$\text{(Rule } \xi_{JP}) \frac{[TB; F, \texttt{PE}_{(u_1|u_2)@d}p; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F, \texttt{PE}_{u_1@d}p \vee \texttt{PE}_{u_2@d}q; L; \texttt{OB}\ X; \texttt{PE}\ Y]}$$

Rules ξ_{DO} and ξ_{DP} decompose the the modal formulae containing delegative composite authorities (in the form of $\texttt{ds}_{(u_1 \rhd u_2)@d}\ p$).

$$\text{(Rule } \xi_{DO}) \frac{[TB; F, \texttt{OB}_{(u_1 \rhd u_2)@d}p; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F, \texttt{OB}_{u_1@d}(\texttt{OB}_{u_2@d}p); L; \texttt{OB}\ X; \texttt{PE}\ Y]}$$
$$\text{(Rule } \xi_{DP}) \frac{[TB; F, \texttt{PE}_{(u_1 \rhd u_2)@d}p; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F, \texttt{PE}_{u_1@d}(\texttt{PE}_{u_2@d}p); L; \texttt{OB}\ X; \texttt{PE}\ Y]}$$

Rules λ_{+U} and λ_{-U} decompose the predicates containing the union of two concepts as one of their arguments. The semantics of predicates shows the philosophy of existing such rules.

$$\text{(Rule } \lambda_{+U}) \frac{[TB; F, p(..., C_1 \sqcup C_2, ...); L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F, p(..., C_1, ...), p(..., C_2, ...); L; \texttt{OB}\ X; \texttt{PE}\ Y]}$$
$$\text{(Rule } \lambda_{-U}) \frac{[TB; F, \neg p(..., C_1 \sqcup C_2, ...); L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F, \neg p(..., C_1, ...) \vee \neg p(..., C_2, ...); L; \texttt{OB}\ X; \texttt{PE}\ Y]}$$

Categorization Rules

The γ rules, categorize the subformulae resulted from the fragmentation rules. γ_O categorize obligation subformulae (of the form $\texttt{OB}_{u@d}p$) into OBF of the set-block. The γ_P categorize permission subformulae (of the form $\texttt{PE}_{u@d}p$) into PEF, and the γ_L rule categorize the literals into L. Note that by literal, we mean a proposition, a predicate, a subdomain relationship, or their negations.

$$\text{(Rule } \gamma_O) \frac{[TB; F, \texttt{OB}_{u@d}p; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F; L; \texttt{OB}\ X, \texttt{OB}_{u@d}p; \texttt{PE}\ Y]}$$
$$\text{(Rule } \gamma_P) \frac{[TB; F, \texttt{PE}_{u@d}p; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F; L; \texttt{OB}\ X; \texttt{PE}\ Y, \texttt{PE}_{u@d}p]}$$
$$\text{(Rule } \gamma_L) \frac{[TB; F, l; L; \texttt{OB}\ X; \texttt{PE}\ Y]}{[TB; F; L, l; \texttt{OB}\ X; \texttt{PE}\ Y]} \qquad (l \text{ is a literal})$$

Sub-Tableaux Construction Rules

In a modal tableaux calculi (like the one we introduce in this chapter), modal formulae specify not only conditions over a world, but also on the worlds which are accessible from it. The κ and θ rules are modal expansion rules. The κ rule is based on this fact that in MA(DL)^2, $\texttt{OB}_{u@d}p$ is true in a world if p is true in its all accessible worlds from u's viewpoint in domain d or its subdomains. Thus, consequence of this rule introduces a new node which holds in a different world from the world in which the rule's assumption holds. In other words, if in an accessible world (here the world of the consequence), the tableaux of formula p closes (which means p is unsatisfiable), in the main tableaux tree, the path ending to it closes, too (because it shows $\texttt{OB}_{u@d}p$ is unsatisfiable).

In the κ rule, we have the same situation. The consequence of this rule, states that a set of formulae X' must be true accompanying with formulae p in at least one new accessible world. Note that by this rule, for each $\texttt{PE}_{u@d}p$ we should create a new node in tableaux tree for the set of formulae $X' \cup p$.

$$\text{(Rule } \theta\text{)}\frac{[TB; \{\}; L, d_1 \preceq d, ..., d_n \preceq d; \texttt{OB}\ X, \texttt{OB}_{u@d_i(1\leq i\leq n)}X'; \texttt{PE}\ Y, \texttt{PE}_{u@d}\{\}]}{[TB; X'; \{\}; \{\}; \{\}]}$$

$$\text{(Rule } \kappa\text{)}\frac{[TB; \{\}; L, d_1 \preceq d, ..., d_n \preceq d; \texttt{OB}\ X, \texttt{OB}_{u@d_i(1\leq i\leq n)}X'; \texttt{PE}\ Y, \texttt{PE}_{u@d}p]}{[TB; X', p; \{\}; \{\}; \{\}]}$$

In these two expansion rules, which are introducing new worlds, in contrast with world-labeling approach, we do not label formulae with the world where they are true (for example we do not need to have $w : p$ for showing that p is true in world w). In fact, we employ auxiliary tableaux approach in which for dealing with formulae holding in new alternate worlds, a (sub)tableaux algorithm is started.

In θ and κ rules, $\texttt{OB}_{u@d_i(1\leq i\leq n)}X'$ includes all formulae $\texttt{OB}_{u@d_i}p_i$ where $d_i \preceq d$. Also, these rules show that in constructing the initial block-set of formulae for subtableaux tree (i.e., the block-set in the consequences of the rules) the set of t-formulae in TB are remained unchanged. This is resulted from the assumption we have in MA(DL)^2 that says ontologies (as a set of t-formulae in $\mathcal{TB}$ of an SKB) describe a shared conceptualization.

The last remaining point about these two rules is that they create a disjunctive branching, which means closing one of the branches results in closing the branching node. Figure 7.1 shows a disjunctive branching created by applying κ rule graphically.

$\mathcal{ALC}$ Tableaux Rules

To infer the subsumption relationships between the concepts in $\mathcal{ALC}$, we used the tableaux rules that are described (and implemented in Prolog) by Herchenröder [80]. We do not mention these rules here.

Closure Rules

The did_1 rule closes a node and consequently, closes all the pathes end to it; due to the existence of a direct clash in the set of literals, i.e., existence of l and its negation $\bar{l}$. Note that this rule is applicable, when the second element in the assumption of the rule is empty. Such a condition is not necessary in theory; however, in practice, it causes to postpone the search for finding a clash to the situation where all formulae are decomposed to its atomic subformulae, and are categorized. Therefore, the performance is increased in this situation.

$$(\text{Rule } did_1)\frac{[TB; \{\}; L, l, \bar{l}; \texttt{OB}\ X; \texttt{PE}\ Y]}{closed}$$

Furthermore, we require the did_2 and did_3 closure rules as described below.

$$(\text{Rule } did_2)\frac{[TB; \{\}; L, \texttt{F}; \texttt{OB}\ X; \texttt{PE}\ Y]}{closed} \qquad (\text{Rule } did_3)\frac{[TB; \{\}; L, \neg\texttt{T}; \texttt{OB}\ X; \texttt{PE}\ Y]}{closed}$$

The iid_1 rule closes a node and all pathes ended to the node, due to the existence of an indirect clash (by subsumption relationships between the concepts). This rule is applicable, when a subsumption relationship $C' \sqsubseteq C$ is inferrable from TB. Inference of subsumption relationships are done based on the $\mathcal{ALC}$ tableaux rules [80].

$$(\text{Rule } iid_1)\frac{[TB \vdash C' \sqsubseteq C; \{\}; L, p(..., C, ...), \neg p(..., C', ...); \texttt{OB}\ X; \texttt{PE}\ Y]}{closed}$$

The iid_2 rule closes a node by existing a clash in the defined subdomain relationships, i.e., existence of $\neg(d \preceq d')$ and ability ro infer $d \preceq d'$ in set L. Inference of subdomain relationships are done based on the reflexivity and transitivity of subdomain relation. The required tableaux rules for this purpose are easy and not described here.

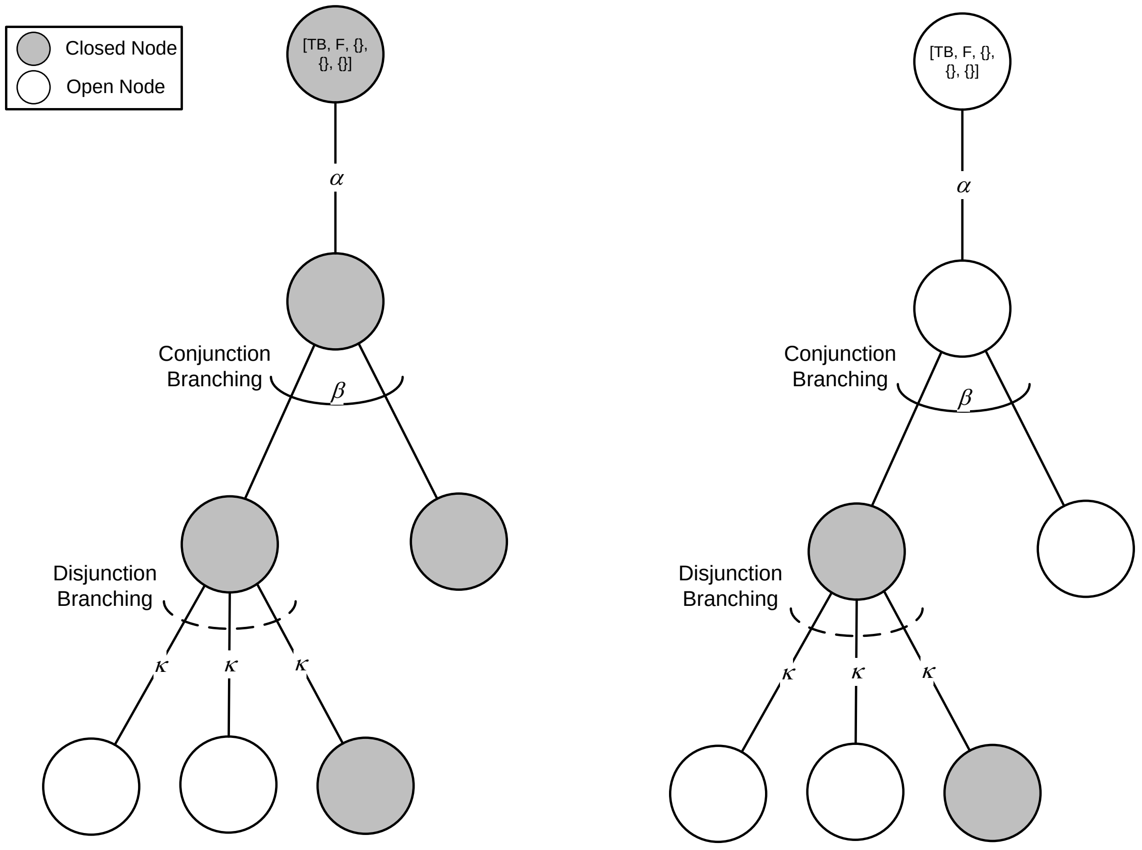


Figure 7.2: The right side is a sample of open tableaux tree, and the left side is a sample of closed tableaux tree.

$$(\text{Rule } iid_2)\frac{[TB;\{\};L,\neg(d\preceq d'),\vdash d\preceq d';\texttt{OB}\ X;\texttt{PE}\ Y]}{closed}$$

Closed Tableaux Tree

The description of tableaux expansion rules for MA(DL)^2 shows that we have two kinds of branching in the tableaux tree. Therefore, in a MA(DL)^2 tableaux tree, a *node* is *closed* in the following situations:

- if it is a leaf, a closure rule is applicable,
- if it is an inner node that is a parent of another node, its child is a closed node,

- if it is a node over a conjunction branching, its all branches are closed,
- if it is a node over a disjunction branching, at least one of its branches are closed.

A *path*, from the root to a leaf, is *closed*, if it does not contain a closed node; otherwise, it is *open*. An MA(DL)^2 tableaux *tree* is *closed*, if it does not contain an open path, or in other words, if its root node is closed; otherwise, it is an *open tree*. Figure 7.2 shows samples of open and closed MA(DL)^2 tableaux tree.

7.1.3 Properties of Tableaux Rules

The three properties, which are investigated for each tableaux algorithm, are termination, soundness, and completeness. In the rest, we prove that the proposed tableaux proof system for MA(DL)^2 logic family has the three properties.

Theorem 7.1.1 (Termination) *The proposed tableaux proof system for MA(DL)^2 logic famuily terminates.*

Proof. To prove the termination of the proposed tableaux, we need to prove that the maximum number of rule applications in the proposed approach is finite. As mentioned earlier, the described rules force four steps in running a tableaux procedure.

- In fragmentation rules, rules α, β, ξ_{JO}, and ξ_{JP} always decompose a formula to its subformulae. Since the length of the initial formula in tableaux procedure is finite, these rules are applicable for a finite number.
- Although the ξ_{DO} and ξ_{DP} rules, do not decrease the length of formulae, they decrease the number of $\triangleright$ operators, which are finite in the initial formula.
- Rules λ_{+U} and λ_{-U} decrease the length of the arguments of the predicates existing finitely in the initial formula.
- The γ categorization rules only categorize the subformulae produced by applying fragmentation rules. Since the number of the subformulae is finite, the number of application of these formulae is finite, too.

- Sub-tableaux construction rules extract subformulae in modal formulae and run a tableaux procedure with shorter length formulae. Since the number of modal operators is finite, the number of applications of these rules is finite, too.
- The termination of tableaux algorithm for inference of subsumptions in $\mathcal{ALC}$ are proved in [80].
- Application of each closure rule, results in closing a path or branch in the tableaux tree.

Following the above discussion, the total number of rule applications in tableaux procedure is finite. Thus, the proposed tableaux procedure terminates. ■

The proof of the soundness of a tableaux system is equal to prove that all its rules preserve the unsatisfiability. In other words, we should prove that in each rule, if the consequence is unsatisfiable, its assumption is unsatisfiable, too.

Theorem 7.1.2 (Soundness of Tableaux) *If there exists a closed tableaux for a c-formula F in MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]_{\&^-}$ logic, then F is unsatisfiable.*

Proof. It is enough to prove that in each rule, if the consequence is unsatisfiable, its assumption is unsatisfiable, too. Then, by induction we can prove that if there exists a closed tableaux tree, formula F is unsatisfiable. Existence of such property in the fragmentation and categorization rules is obvious. Considering the semantics of MA(DL)^2, existence of this property in sub-tableaux construction rules (i.e., rules θ and κ) can be proved easily. For example, it is clear that if X in one of the accessible world is unsatisfiable, $\texttt{OB}_{u@d}X$ is unsatisfiable as well. The soundness of the tableaux rule for $\mathcal{ALC}$ is presented in [80]. In the closure rules, the consequences are closed, due to the existence of direct or indirect clash in the assumptions (showing unsatisfiability). Therefore, by existing a closed tableaux tree for a formula F, by stepping backward from the leaves (which are closed) to the root, we conclude that the formula in the root (i.e., formula F) is unsatisfiable. ■

The completeness of a tableaux system, means for each c-formula in MA(DL)$^2[^{\mathcal{U}}_{\mathcal{D}}]_{\&^-}$, there exists a closed tableaux tree. The proof sketch of the completeness theorem is presented in the rest after proving a lemma.

Lemma 7.1.1 *Let F be a c-formula in $MA(DL)^2[^{\mathcal{U}}_{\mathcal{D}}]_{\&^-}$ in negation normal form (NNF), such that does not contain any modal subformula (or modal statuses* `OB` *and* `PE`*). If T is a tableaux tree resulted from applying the tableaux rules (except θ and κ, which are not applicable), and contains an open path, formula F is satisfiable.*

Proof. Since in the proposed tableaux system, we used set-labeling method, each leaf contains all the subformulae of the formulae exist in the inner nodes of the tree from the leaf to the last branching node. Therefore, in the open path, the leaf node is in the form $[TB; \{\}; L; \{\}; \{\}]$ (w.r.t. the fact that there is no modal status in F), where L contains all atomic subformulae of the inner nodes to the last branching node. In the tableaux rules (except θ and κ) only the β rule creates a conjunctive branching. If we suppose β is n times applicable, we can write F as $F = F_1 \vee ... \vee F_n$ such that the open path contains subformula F_1. Therefore, L contains all subformulae of F_1. Since the path is open, L is a consistent set (there is no clash in it), and hence, F_1 and F are satisfiable. ∎

Theorem 7.1.3 (Completeness of Tableaux) *If F is unsatisfiable in $MA(DL)^2[^{\mathcal{U}}_{\mathcal{D}}]_{\&^-}$ logic, then there exists a closed tableaux for F.*

Proof. Considering the termination of the proposed tableaux method for MA(DL)^2 logic family, we assume T_n as a finite tableaux tree obtained from formula F by applying all possible expansion rules. Now, we prove by induction that if T_n is not closed, then F is satisfiable, which contradicts the assumption of the theorem.

If T_n is not closed, it contains a subtree in which none of the paths are closed, and if it contains a disjunction branching node of T_n, it contains all the branches from that node as well.

1. If F does not contain modal subformulae (i.e., with `OB` and `PE` operators), the open subtree is only an open path. In this case, by Lemma 7.1.1, formula F is satisfiable, which contradicts the assumption.

2. If there exist modal subformulae in F, certainly θ and κ rules are applied for constructing T_n. Considering the preconditions of these two rules, it is clear that they can be applied after applying α, β, ξ, and γ rules. Thus, before applying the θ and κ rules, we can just have conjunctive branching (by applying the β rule). Note that in subtrees

resulted from applying the θ and κ rules, it is possible to apply the α, β, ξ, and γ rules. Similar to the proof of Lemma 7.1.1, we can write formula F as $F = F_1 \vee ... \vee F_n$. Since, we assumed that T_n is open, the branch of one the F_i's should be open. If F_i contains modal operators, in the open subtree, we have a disjunction branching resulted from applying the θ and κ rules, and its all branches are open.

- If the κ rule by existing $\mathtt{OB}_{u@d_i(1\leq i\leq n)}\ X'$ is applied, each resulted branch is a tableaux tree. If the created subtableaux tree does not contain modal operators `OB` and `PE`, by previous case, X', p are consistent and satisfiable.
- If the θ rule is applied, openness of the all branches means we can easily construct a model in which set `OB` X of formulae is consistent.

If any branch in the tree contains a modal subformula, we can continue by induction and get the same result, i.e., the consistency of sets `OB` X and `PE` Y of formulae.

In each of the above cases, since L in the set-block of the branching node in consistent (because no closure rule is applied) and the consistency of `OB` X and `PE` Y, we conclude that F_i is valid. Thus, F is valid, too. By applying the θ rule, we have the same proof.

Therefor, by assuming that T_n is open, formula F is valid, which contradicts the assumption of the theorem. Thus, the theorem holds.

Hence, the assumption that there exists an open path in T_n cannot hold and a closed tableaux tree exists for formula F. ■

7.1.4 Implementation in Prolog

For ensuring the correctness and applicability of the proposed tableaux system for MA(DL)^2 logic in practice, the tableaux rules are implemented in Prolog. Prolog is a declarative language with backtracking execution strategy. Efficient implementation of unification and goal-oriented search strategy, makes this language a proper choice for implementing tableaux algorithms.

To use the implemented inference engine, terminological box $\mathcal{TB}$ and security rule box $\mathcal{SB}$ should be defined as follows.

```
assert(tbox([comma separated list of t-formulae])).
assert(sbox([comma separated list of c-formulae])).
```

Table 7.1: Syntax of formulae in MA(DL)^2 inference engine implemented in Prolog.

MA(DL)^2 Formula	**Syntax in the Engine**	**MA(DL)^2 Formula**	**Syntax in the Engine**
$c_1 \equiv c_2$	`equiv(c1,c2)`	$p(c_1, ..., c_n)$	`pred(p, [c1,...,cn])`
$c_1 \sqsubseteq c_2$	`subsum(c1,c2)`	$p \wedge q$	`p ^ q`
$c_1 \sqcup c_2$	`or(c1,c2)`	$p \vee q$	`p v q`
$c_1 \sqcap c_2$	`and(c1,c2)`	$\neg p$	`-p`
$\neg c$	`~c`	$p \rightarrow q$	`p=>q`
$\forall r.c$	`forall(r,c)`	$p \leftrightarrow q$	`p<=>q`
$\exists r.c$	`exist(r,c)`	$\texttt{OB}_{u@d}p$	`ob(u@d,p)`
$u_1 \| u_2$	`u1#u2`	$\texttt{PE}_{u@d}p$	`pe(u@d,p)`
$u_1 \rhd u_2$	`u1>u2`	$\texttt{IM}_{u@d}p$	`im(u@d,p)`
$d_1 \preceq d_2$	`subdomain(d1,d2)`	$\texttt{GR}_{u@d}p$	`gr(u@d,p)`

For checking the validity (or proof) of c-formula F, we should call `proof(F)`. The syntax of MA(DL)^2 logic in Prolog is shown in Table 7.1. The Prolog code of version 2.07 of the MA(DL)^2 inference engine can be find in Appendix B.

7.1.5 Evaluation and Experimental Results

The MA(DL)^2 inference engine, which is implemented in Prolog, is used to evaluate the time complexity of inference in MA(DL)^2 logic family, in practice. To this aim, the program was loaded in SWI-Prolog interpreter and was run on a PC with Intel Core2 Dou 3.00 GHz processor, 3 GB RAM, and Windows XP operating system. All tests run times (inference times) were measured using *Statistics/2* predicate in SWI-Prolog.

In evaluating the MA(DL)^2 inference engine, two parameters, i.e., the complexity of t-formulae, and the length of c-formulae (based on the number of modal operators), are taken as variables in the different prepared test cases.

Evaluation based on the Complexity of T-Formulae

For this purpose, the Extended-Mindswap test cases are used. These test cases are the extension of the Mindswap test cases, and they were prepared by Herchenröder [80] for evaluating $\mathcal{ALC}$ inference engines. Mindswap test cases were initially produced by the group working on the Mindswap project at University of Maryland[2]. In Extended-Mindswap test

[2]For more details about the Mindswap project and access to the prepared test cases, you can visit `http://www.mindswap.org`.

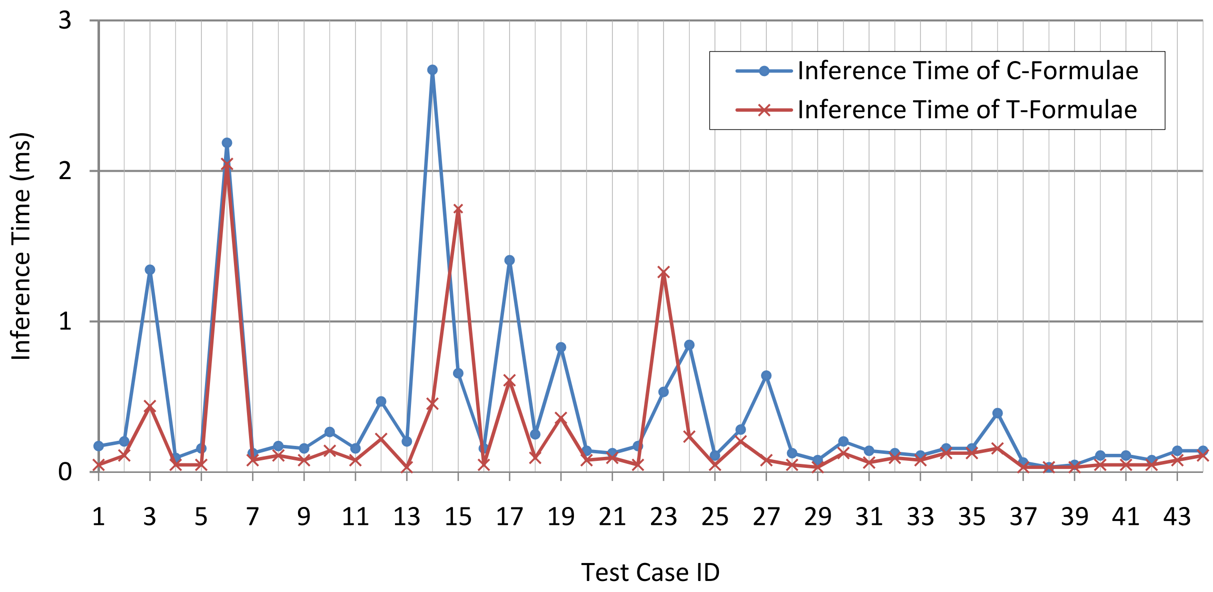


Figure 7.3: Inference time of t-formulae and produced c-formulae in MA(DL)^2 based on the Extended-Mindswap test cases.

cases, there are 44 queries such that each of them has a special complexity or focuses on the special properties.

To leverage such test cases for evaluating the MA(DL)^2 inference engine, for each test case, the set of formulae taken for TBox are added to $\mathcal{TB}$ in the SKB. $\mathcal{SB}$ in the SKB is filled based on the atomic subformulae of the query in the test case as mentioned here. For each atomic subformulae of the query[3] like C_i, a formula $\texttt{OB}_{u@d}p(C_i)$ is added to $\mathcal{SB}$. Finally, the query for evaluating the MA(DL)^2 inference engine in the resulted SKB, is $\texttt{PE}_{u@d}p(C)$, where C is the query in the Extended-Mindswap test case. The set of test cases created from the Extended Mindswap test cases by the above method are presented in Appendix C. The results of the evaluation of the inference engine based on the prepared test cases are shown in Figure 7.3.

Evaluation based on the Size of the SKB

The impact of the size of the SKB on the inference time has been evaluated by generating the content of the SKB by different sizes randomly. For each generated case, the inference time

[3]Note that a query in a Mindswap test case is a description of a complex concept in $\mathcal{ALC}$, which its satisfiability should be inferred.

in the worst case (i.e., the inference time of for an unsatisfiable formula) has been measured. Note that in a tableaux method, the inference of an unsatisfiable formula using refutation approach, results in constructing the whole tableaux tree (for finding a possible close tree).

The size of the SKB in this evaluation, is taken as a function of the number of `OB` modal formulae, the number of `PE` modal formulae, and the number of subsumption relationships. The description of generated test cases, as well as the results of the evaluations are as follows. The reported inference time in each test, is the average of 100 runs. The evaluated response time is the aggregation of the time required for the all required conversions (e.g., to NNF, DNF, and NNF), the time of constructing the required data structures, the time of building the tableaux tree, and the time of freeing the memory occupied by the data structures.

Evaluation with Variable Number of `OB` Modal Formulae: The fixed parameters in this case are as follows:

Number of concepts= 20	Number of subsumption relationships= 10
Number of conceptual predicates= 5	Number of authorities= 10
Number of security domains= 1	Number of `PE` modal formulae= 20

The results are shown in Figure 7.4. The diagram shows that by keeping other parameters unchanged, the inference time in MA(DL)^2 inference engine is a linear function of the number of `OB` modal formulae.

Evaluation with Variable Number of `PE` Modal Formulae: The fixed parameters in this case are as follows:

Number of concepts= 20	Number of subsumption relationships= 10
Number of conceptual predicates= 5	Number of authorities= 10
Number of security domains= 1	Number of `OB` modal formulae= 20

The results are shown in Figure 7.5. The diagram shows that similar to the previous case, by keeping other parameters unchanged, the inference time in MA(DL)^2 inference engine is a linear function of the number of `PE` modal formulae.

Evaluation with Variable Number of `OB` and `PE` Modal Formulae: The fixed parameters in this case are as follows:

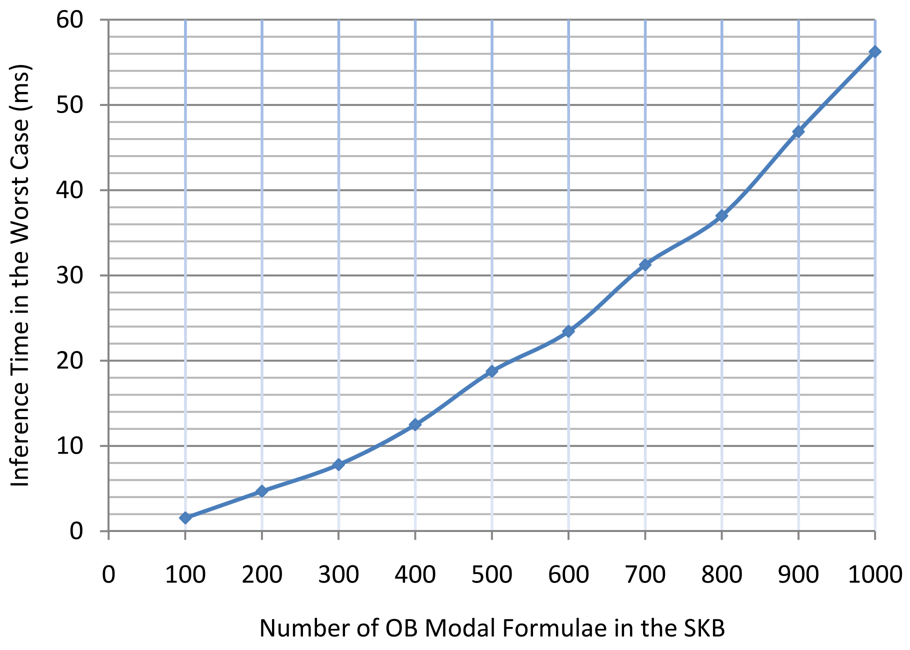


Figure 7.4: Evaluation of inference time in MA(DL)^2 inference engine with variable number of `OB` modal formulae.

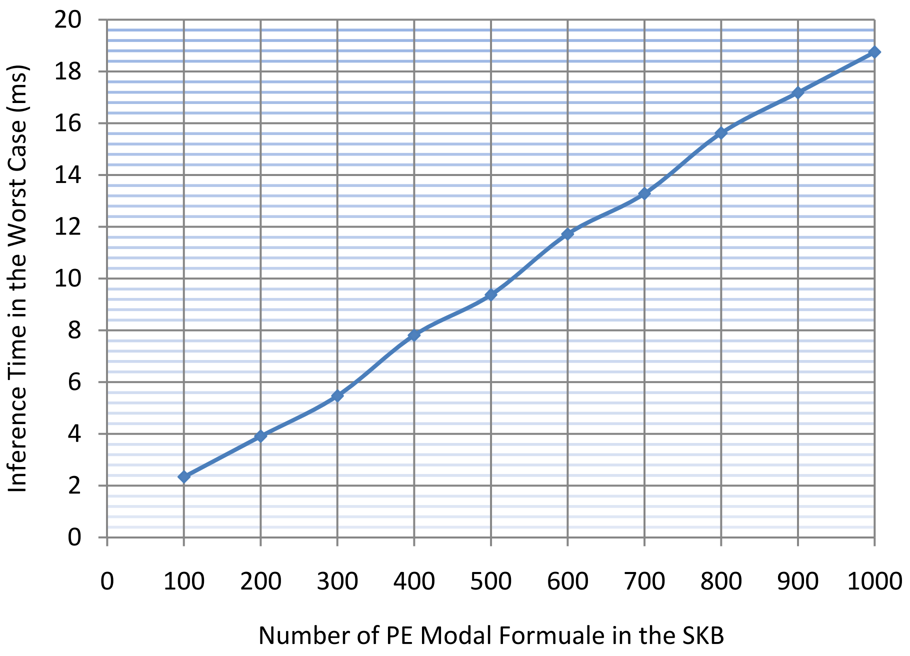


Figure 7.5: Evaluation of inference time in MA(DL)^2 inference engine with variable number of `PE` modal formulae.

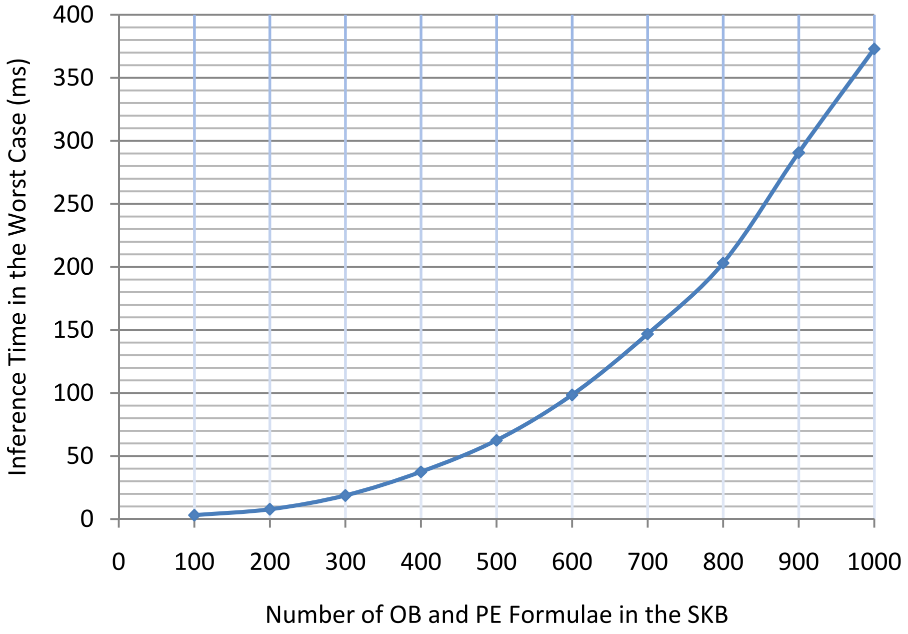


Figure 7.6: Evaluation of inference time in MA(DL)^2 inference engine with variable number of `OB` and `PE` modal formulae.

Number of concepts= 20	Number of subsumption relationships= 10
Number of conceptual predicates= 5	Number of authorities= 10
Number of security domains= 1	

The results are shown in Figure 7.6. The diagram shows that the inference time is polynomial of order 2 in this case. Since in the implemented inference engine, in the worst case, for each `PE` formula, all the `OB` formulae are visited, by having n number of `OB` and `PE` formulae, the time complexity of inference is $O\left((\frac{n}{2})^2\right)$.

<u>Evaluation with Variable Number of Subsumption Relationships:</u> The fixed parameters in this case are as follows:

Number of conceptual predicates= 5	Number of authorities= 10
Number of security domains= 1	Number of `OB` modal formulae= 20
Number of `PE` modal formulae= 20	

In this case, the number of subsumption relationships and the number of concepts (participating in these relationships) are variable. The number of concepts are twice the number of relationships. The results of the evaluation of the system in this case are shown in Figure

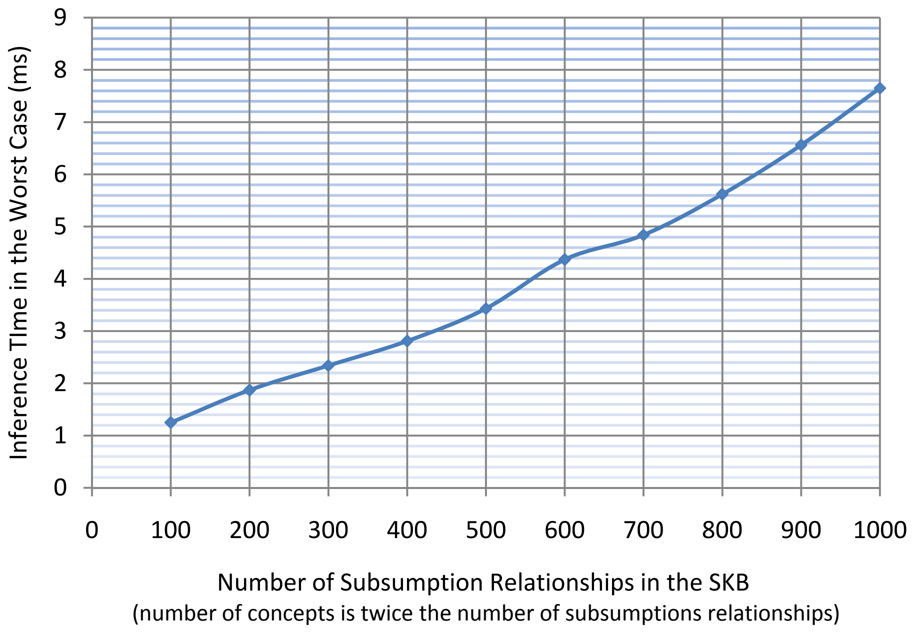


Figure 7.7: Evaluation of inference time in MA(DL)^2 inference engine with variable number of subsumption relationships (and concepts).

7.7. Comparing the measured inference times in this case with the values obtained in the previous cases shows that modal formulae has more significant overhead in the inference engine. The main reason is constructing subtableaux trees by the expansion rules of modal formulae in the proposed tableaux system.

7.2 Security Agent Prototype

A prototype of a security agent, which enforces the security policies based on the MA(DL)^2-AM authorization model, is implemented in this research. In this prototype, we tried to use the tools and components that are provided for semantic-aware environments. The source code of the security agent prototype is available at `http://ce.sharif.edu/~m_amini/madl2/prototype` under GNU GPL license.

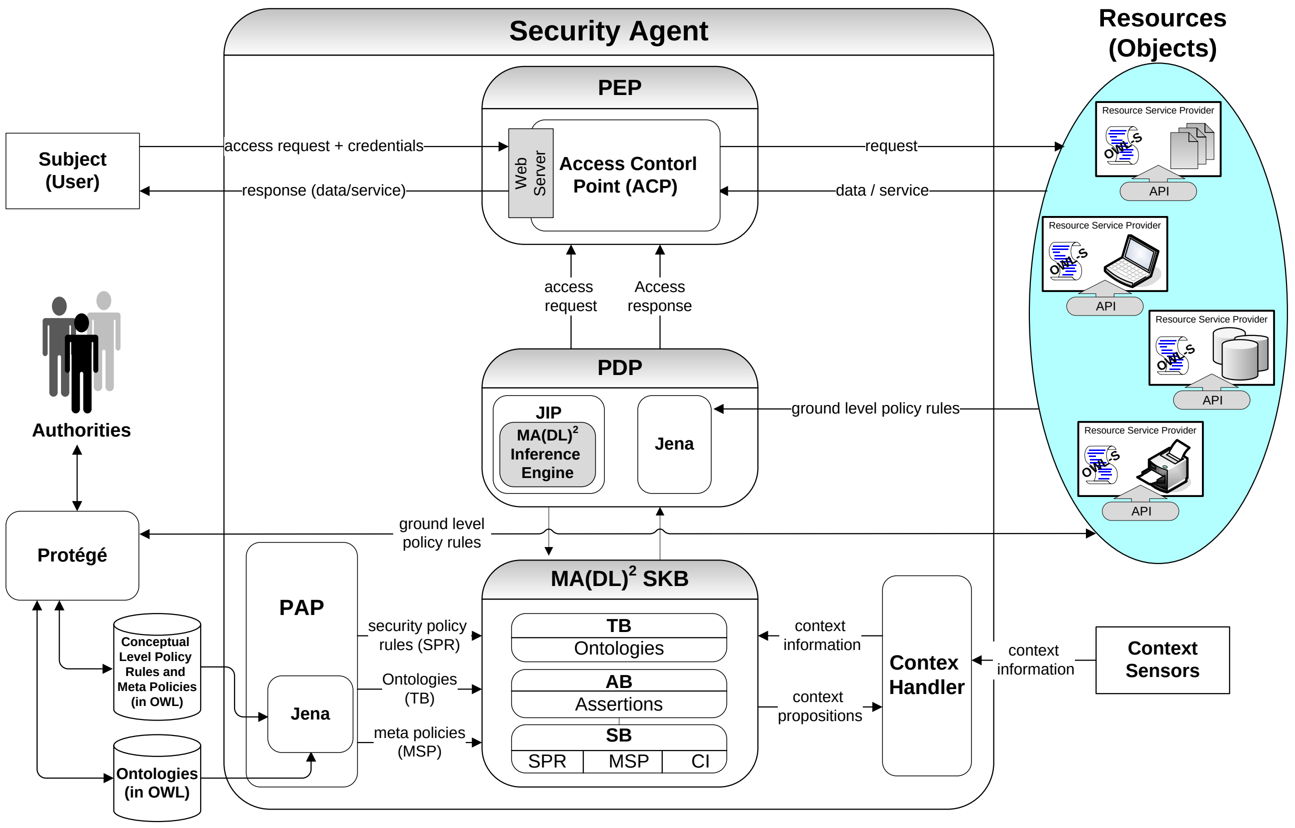


Figure 7.8: The master design of the implemented security agent prototype.

7.2.1 Master Design

A prototype of a security agent based on the architecture shown in Figure 3.5 is implemented using Google Web Toolkit (GWT[4]) on Java platform. The important characteristics of the current version of this prototype are as follows.

- We can use Protégé[5] for specifying the ontologies of subjects, objects (resources), and actions. In fact, Protégé plays the role of the UI for PAP (Policy Administration Point) component in the security agent. The ontologies written in Protégé are stored in OWL files, which are read by the security agent.

- Security policy rules and also meta-policies are specified in Protégé, too. They are stored in OWL files.

[4]Available at `http://code.google.com/webtoolkit`.

[5]Available at `http://protege.stanford.edu`.

- The ground (individual) level policy rules of resources (objects) are specified in OWL-S in their related Web services, based on the model described in Section 3.4.3.
- Jena[6] framework is used to read OWL and OWL-S files. The conceptual level and ground level policy rules are read by Jena and are written in the MA(DL)^2 SKB. The inferences over $\mathcal{TB}$ (for subsumption relationships) and $\mathcal{AB}$ (for most specific concepts, i.e., `msc`, and instance checking) are also done by Jena.
- The MA(DL)^2 inference engine (implemented in Prolog) is employed by the JIP[7] profiling tool for policy inference in PDP.
- In this prototype, the access requests can be received from a Web user interface and access decision is made based on the procedure described in Section 6.3.3; however, the obligations cannot be enforced in this implementation. Obligation enforcement is itself an interesting research field, whose some approaches can be found in [109, 26, 62, 118].

Based on the above characteristics, the master design of the implemented security agent prototype is shown in Figure 7.8.

7.2.2 Experimental Results

To evaluate the current implementation of the proposed framework, some test cases were run on a system with 1.66 GHz Core2Duo CPU and 2 GB of RAM. Note that the current implementation is not perfect and its performance can be improved in different ways.

In figures 7.9 and 7.10, the number of policy rules (in both conceptual and ground levels) has a near to linear relationship with the number of the concepts. As depicted in the diagrams, by increasing the number of the concepts and policy rules, the response time increases in a very slight exponential manner, near to linear. This means that the agent decides and responses in an acceptable time, in practice. In Figure 7.11, the response time with different number of concepts is shown. In this diagram, one conceptual level policy rule beside a ground level policy rule exist in the SKB. In this diagram, we just consider the worst case.

[6]Available at `http://jena.sourceforge.net`.

[7]Available at `http://jiprof.sourceforge.net`.

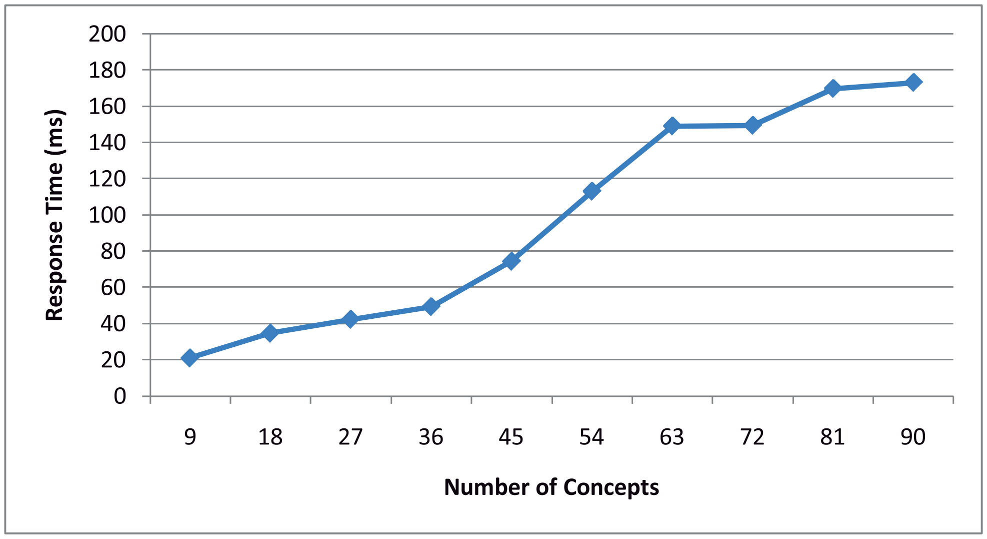


Figure 7.9: Response time of granting an access vs. the number of concepts, while policy rules increase relative to the number of concepts.

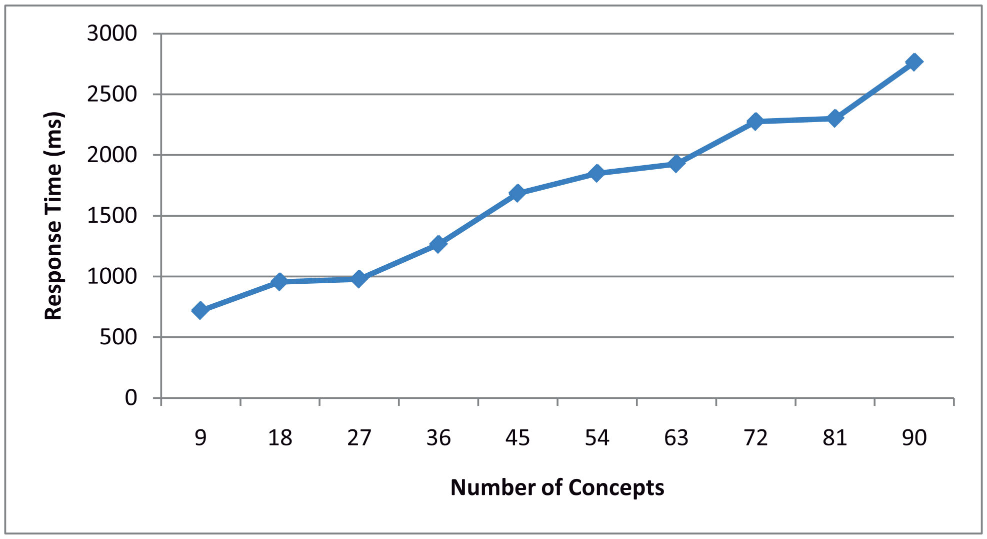


Figure 7.10: Response time of denying an access vs. the number of concepts, while policy rules increase relative to the number of concepts.

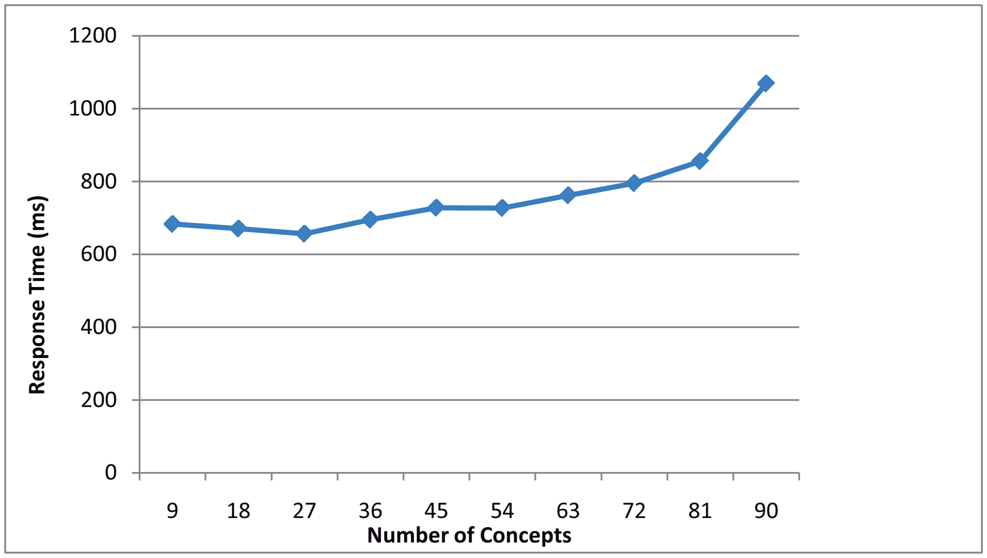


Figure 7.11: Response time of denying an access vs. the number of concepts, while there is only a single policy rule.

Comparing the three diagrams show that the size of $\mathcal{SB}$ in the SKB has more effect than the size of $\mathcal{TB}$ on the response time of the agent. Note that when the size of $\mathcal{SB}$ increases (by adding the random policy rules) randomly, the possibility of the conflicts between the policy rules and therefore the response time increase as well.

7.3 Summary

A practical access control system based on the proposed authorization model can show the applicability of the proposed model. for this purpose, since the model is based on the MA(DL)^2 logic family, we require to develop an inference engine for MA(DL)^2. The proof system introduced in Chapter 4 is an axiomatic system, which cannot be automated in practice. Thus, in this chapter, an analytic tableaux approach for MA(DL)^2 logic family is proposed, and also implemented in Prolog. The implemented MA(DL)^2 inference engine was evaluated and the results are presented in this chapter.

Using the MA(DL)^2 inference engine, a security agent prototype, based on the MA(DL)^2-AM authorization model has been implemented. The master design of this prototype and experimental results are presented in this chapter. The results show that although the logical inference is very time-consuming in theory, in this application, its cost is acceptable in practice. In future work described in Chapter 8, some approaches for decreasing the response time in the access control systems developed (based on the MA(DL)$^{@}$-AM authorization model) are introduced.

Chapter 8

Conclusions and Future work

The shift from the current computational environments to the semantic aware environments provides a basis for interoperability and increasing the capability of machines in processing and interpreting the information. Characteristics of such semantic-aware environments suppress using the traditional and classical authorization models for them. Also, the main drawback of most of the new models presented specifically for SAEs, is that they do not have a trustable consistent set of rules for policy inference based on the semantic relationships defined in the abstract conceptual layer. The problem which could be tackled by logics in such models.

The most important advantages of using logic for authorization in semantic-aware environments are the abstraction of logic (enables composition of security policies for heterogeneous distributed environments), inference ability (enables inferring implicit policy rules from the explicit ones), expressiveness (enables specifying different types of policies), and providing a basis for ensuring the soundness of the model in policy derivation. In this thesis, a logic family, called MA(DL)^2, and an authorization model based on this logic, called MA(DL)^2-AM, have been proposed for policy specification and inference in SAEs. Using MA(DL)^2 logic in MA(DL)^2-AM model, ensures that the set of rules employed for implicit policy derivation (due to the semantic relationships exist in the abstract layer) would be sound, consistent, and complete. Such a guarantee is the result of existing a formal semantics for the proposed logical language, and proving the soundness, completeness, and decidability of its proof system. This property, is one of the outstanding characteristics of MA(DL)^2-AM in comparison with other authorization models proposed for SAEs.

MA(DL)^2 logic, which is introduced in this thesis, is in fact a normative logic family, whose core is resulted from the integrating description logic with n-ary predicates on concepts, and multi-authority (poly-modal) version of standard deontic logic. Using Core MA(DL)^2 logic, we can express explicit norms on concepts and infer the implicit norms based on the defined semantic relationships in ontologies. This is the answer to the main requirement that we have for authorization in conceptual level in SAEs. Extending Core MA(DL)^2 by logic of composite authorities (employed for cooperative security administration in shared subdomains) results in MA(DL)$^2[\genfrac{}{}{0pt}{}{\mathcal{U}}{-}]$, and by logic of security domains results in MA(DL)$^2[\genfrac{}{}{0pt}{}{-}{\mathcal{D}}]$ in this family. MA(DL)$^2[\genfrac{}{}{0pt}{}{\mathcal{U}}{\mathcal{D}}]$ in this logic family has the most expressive power and has all the capabilities of the other members.

Considering the Kripke semantics presented for the members of MA(DL)^2 logic family, the soundness of them has been proved. However, the completeness of some of them are proved by applying some constraints in the proposed logics. Although MA(DL)^2 logic has been proposed for authorization in SAEs, the expressive power of the proposed logic besides its proved soundness, completeness, and decidability properties make it useful for using in other applications in computer science. Samples of these applications are specification and inference of the behavior of the mobile agents, and composition of Web services.

MA(DL)^2-AM is the authorization model proposed in this thesis, based on MA(DL)^2 logic family, for access and obligation policy specification, composition, and inference in SAEs in both conceptual (abstract) and ground (individual or resource) levels. The policies can be specified distributed, by different authorities in different security domains. Preserving the consistency of the policy rules in the security knowledge base is the principle considered in this model in adding, updating, and removing policy rules, by using the inference ability that MA(DL)^2 gives to the model. All the possible conflicts in this model are also surveyed and the solutions required to solve or prevent them are fed to the model. Evaluating the capabilities of the proposed model shows that MA(DL)^2-AM is an attribute-based (credential-based) model, which capables us to specify discretionary, mandatory, and role-based policies. Such expressive power is the result of the abstraction level of the model and existence of normative statuses, i.e., permission, prohibition, obligation, and gratuitousness.

In order to use MA(DL)^2 logic in real applications (in our case, in access control systems), we require to have a automated inference engine. Since the Hilbert style proof theory, which is presented in this thesis for MA(DL)^2, cannot be automated, an analytic tableaux method has been proposed for this purpose. The termination, soundness, and completeness of the proposed tableaux method have been proved, and its expansion rules have been implemented in Prolog. Using the MA(DL)^2 inference engine, a prototype of a security agent for access control based on the proposed access control procedure in MA(DL)^2-AM has been developed and the experimental results have been reported in this thesis.

Using logics (especially modal logics) in different applications of computer science has many advantages. In this thesis, one of its usage, i.e., for authorization in SAEs, has been targeted. Proposing a new logic, or adapting or combining existing logics for a special purpose is a difficult task. Regarding this fact, to make this thesis more educational, we presented MA(DL)^2 logic family, step by step, and in each step, we focused on the effects of the new features added to the logic. The overall process, which is taken in this thesis, can be used as a pattern for developing and using logics in other applications of computer science. The steps of this overall process is requirement analysis, designing the required logic by presenting syntax, semantics, and proof theory (and step by step from the core requirements), proving the required logical and computational properties (such as soundness, completeness, and computational complexity) and applying the required limitations to obtain the proper properties, using the logic in the desired application, and finally presenting an automated reasoning approach to make the proposed logic implementable in that application.

8.1 Future Work

Although MA(DL)^2-AM authorization model has many advantages, and satisfies the main security requirements of SAEs, we can increase the expressive power of this model by enhancing the capabilities of its logical language, i.e., MA(DL)^2. The drawback of this model is that the specification of exceptions is impossible in this model. Also the default access policy cannot be specified using its logical policy specification language. Thus, we cannot participate the default access policy in automated policy rule inference. Such weaknesses can be solved by proposing a non-monotonic version of this logic, which has a paraconsistent semantics. Note that we foremost, need to survey that how much is useful to have exceptions in this model, and how much complexity will be imposed by adding such a feature. Worthwhile to note that existing semantic relationships (in conceptual abstract layer) might make specifying and handling the exceptions so much complicated and useless.

In designing MA(DL)^2 logic family, we used $\mathcal{ALC}$ description logic as a basis. Thus, in specification of subjects, objects (resources), and actions ontologies we are limited to the expressive power of $\mathcal{ALC}$. Whereas nowadays, more powerful variants of description logics are being used for description of ontologies in the abstract conceptual layer of SAEs. Therefore, we can extend MA(DL)^2 by employing more expressive description logics in its core. For this purpose, the effect of the new constructors (for complex concept definition) on the inference of n-ary predicates should be considered in the first step.

Moreover than extending the logic, we can enumerate some research topics as future work related to the application of MA(DL)^2 logic and MA(DL)^2-AM authorization model. One of the main issues in this regard is the access control mechanisms that can be designed based on MA(DL)^2-AM. The prototype introduced in this thesis is one of the possible access control mechanisms implemented based on this model. The important problem in logic-based access control systems (mechanisms) is the low performance (long response time) of these systems. There are some techniques that can be used to propose more efficient access control systems based on the MA(DL)^2-AM model, although in this application (i.e., access control), in real environments, we have a low number of non-complicated policy rules, which do not make policy inference so much time-consuming in practice. Some of the applicable techniques for this purpose are as follows.

- The MA(DL)^2 inference engine is implemented using analytic tableaux approach. The tableaux systems have clear opportunities for parallelization, which significantly decreases the inference time [92, 106] (The different branches of the tableaux proof tree can be processed in parallel). Parallelization of the proposed tableaux for policy inference can have a great effect on decreasing the response time. This research topic becomes more important by considering the widespread usage of multi-core CPUs in new computer systems.

- One of the major techniques in logical systems is *client proof carrying* approach. In this approach, the requester (subject) provides some of the proofs that the security agent requires (e.g., most specific concepts or the authorization rules that shows the requester can access the requested resource), and the security agent just needs to verify the proof (which has a very low cost). In fact, in this technique, requesters sustain part of the inference cost.

The last point worthwhile to note is that MA(DL)^2-AM is a general authorization model for the environments that are using semantic technology for providing an abstract meta-data layer. Therefore, we may customize or simplify the model for special environments where special requirements exist or semantic technology is used in a restricted way. Thus, in the customized or simplified version of MA(DL)^2-AM, we may reach more efficient access control procedure.

Appendix A

Case Studies

The MA(DL)^2-AM authorization model, which is introduced in this thesis, can be leveraged in different kinds of semantic-aware environments. Two case studies describing the usage of this model are presented in this appendix. Note that in describing these case studies, we focus more on how to specify the authorization policies in them.

A.1 Comprehensive Election System

The comprehensive election system (ES) works in a distributed manner and includes Presidential Election System (PES), Mayor Election System (MES) and Parliament Election System (LES). Each type of election system has instances executed in a city or village based on its functionality and people of that area can use its services through the web or exclusive intranet.

A.1.1 Fundamental Elements

In the following, you can see the description of the ontology of resources (objects) that should be protected in our system. Note that the concepts like PES_i describing the individuals of election subsystems for cities or villages.

$\text{ES} \equiv \text{PES} \sqcup \text{MES} \sqcup \text{LES}$

$\text{PES} \equiv \text{PES}_{c_1} \sqcup \text{PES}_{c_2} \sqcup \text{PES}_{v_1} \sqcup \text{PES}_{v_2} \sqcup \text{PES}_{v_3}$

$\text{MES} \equiv \text{MES}_{c_1} \sqcup \text{MES}_{c_2}$

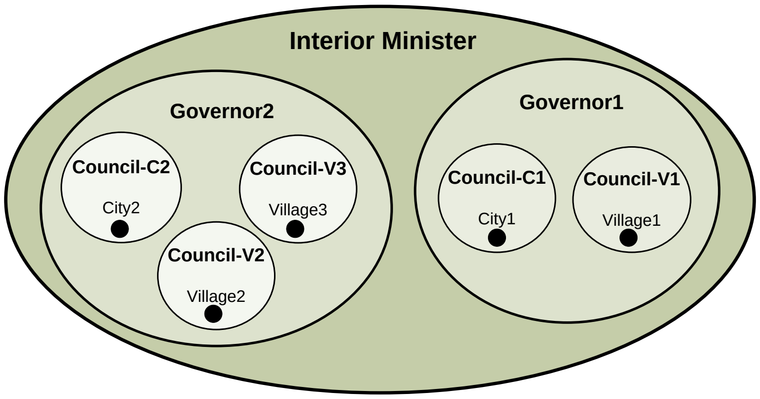


Figure A.1: Authorities structure and their domains.

$\mathrm{LES} \equiv \mathrm{LES}_{c_1} \sqcup \mathrm{LES}_{c_2} \sqcup \mathrm{LES}_{v_1} \sqcup \mathrm{LES}_{v_2} \sqcup \mathrm{LES}_{v_3}$
$\mathrm{ES}_1 \equiv \mathrm{ES}_{c_1} \sqcup \mathrm{ES}_{v_1}$
$\mathrm{ES}_{c_1} \equiv \mathrm{LES}_{c_1} \sqcup \mathrm{MES}_{c_1} \sqcup \mathrm{PES}_{c_1}$
$\mathrm{ES}_{v_1} \equiv \mathrm{LES}_{v_1} \sqcup \mathrm{PES}_{v_1}$

As a result of regionalism, each city or village has an *election council* that manages and controls the elections. Each city with its surrounding villages are under administration of a *governor* who is the authority of all election subsystems in them. The election system (ES) is under administration of *interior minister*. In Figure A.1, the structure of authorities and their domains are represented.

The set of action concepts, which are possible to be requested on the election systems, is defined as $A = \{\text{Vote, Count, RegisterCandidate,SeeResults}\}$.

The ontology of subjects is represented in Figure A.2 and specified in description logic as follows. Note that the system is based on credentials and each user or agent for proving

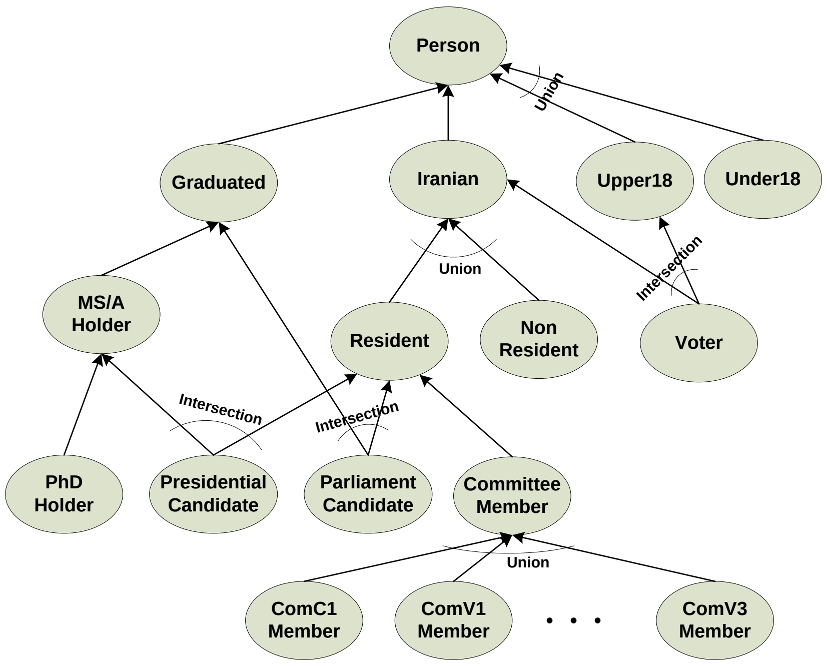


Figure A.2: Subjects ontology of election system.

his/her subjective role in the system, provides a special credential.

$$\text{Iranian} \sqsubseteq \text{Person} \qquad \text{Upper18} \sqsubseteq \text{Person} \qquad \text{Under18} \equiv \text{Person} \sqcap \neg\text{Upper18}$$

$$\text{Iranian} \equiv \text{Resident} \sqcup \text{NonResident} \qquad \text{Graduated} \sqsubseteq \text{Person}$$

$$\text{MS/A Holder} \sqsubseteq \text{Graduated} \qquad \text{PhD Holder} \sqsubseteq \text{MS/A Holder}$$

$$\text{PresidentialCandidate} \equiv \text{MS/A Holder} \sqcap \text{Resident}$$

$$\text{ParliamentCandidate} \equiv \text{Graduated} \sqcap \text{Resident}$$

$$\text{Voter} \equiv \text{Iranian} \sqcap \text{Upper18}$$

$$\text{CommitteeMember} \sqsubseteq \text{Resident}$$

$$\text{CommitteeMember} \equiv \text{ComC1Member} \sqcup \text{ComV1Member} \sqcup ... \sqcup \text{ComV3Member}$$

The set of context propositions that are used to describe the policy rules in the system are as follows.

$$\begin{aligned}
CX &= \{cx_0, cx_1, cx_2, cx_3, cx_4, cx_5, cx_6\}\\
cx_0 &= \text{time is 8am - 18pm}\\
cx_1 &= \text{date is 2008/5/21}\\
cx_2 &= \text{date is 2008/3/15 - 2008/3/31}\\
cx_3 &= \text{time is after 21pm of 2008/5/21}\\
cx_4 &= \text{date is 2008/3/25 - 2008/4/5}\\
cx_5 &= \text{time is after 19pm of 2008/5/21}\\
cx_6 &= \text{time is between 18pm 2008/5/21 and 18pm 2008/5/22}
\end{aligned}$$

A.1.2 Sample Policy Rules

The sample policy rules of the system are defined as follows.

1. Everybody who is resident in the country and has at least MS/A degree can register as a presidential candidate with any presidential election system, during 2008/3/15 to 2008/3/31.

$$cx_2 \rightarrow \mathtt{PE}_{\text{InteriorMinister}}\boldsymbol{do}(\text{MS/A Holder} \sqcap \text{Resident}, \text{PES}, \text{ RegisterCandidate})$$

2. Everybody who is graduated and resident in the country, during the 2008/3/25 to 2008/4/5, can register as a parliament candidate with any parliament election system.

$$cx_4 \rightarrow \mathtt{PE}_{\text{InteriorMinister}}\ \boldsymbol{do}(\text{Graduated} \sqcap \text{Resident}, \text{LES}, \text{ RegisterCandidate})$$

3. The presidential and parliament election will be held in 2008/5/21, between 8am to 18pm and a voter must be Iranian and at least 18 years old.

$$cx_0 \wedge cx_1 \rightarrow \mathtt{PE}_{\text{InteriorMinister}}\boldsymbol{do}(\text{Iranian} \sqcap \text{Upper18}, \text{ES}, \text{ Vote})$$

4. Counting the votes in cities must be started at most 3 hours and in villages 1 hour after the voting deadline by the committee selected by each city's or village's election

council. This policy rule, for City1 and Village1 can be stated as follows.

$$cx_3 \rightarrow \texttt{OB}_{\text{Council-C1}}\ \boldsymbol{do}(\text{ComC1Member}, \text{PES}_{c_1} \sqcup \text{LES}_{c_1} \sqcup \text{MES}_{c_1}, \text{Count})$$
$$cx_5 \rightarrow \texttt{OB}_{\text{Council-V1}}\ \boldsymbol{do}(\text{ComV1Member}, \text{PES}_{v_1} \sqcup \text{LES}_{v_1}, \text{Count})$$

5. Following Interior Minister's statement, everybody can see the results of elections, 3 hours after finishing the elections. However, an authority of an area (domain) can restrict this access. Some sample policy rules are as follows.

$$cx_3 \rightarrow \texttt{PE}_{\text{InteriorMinister}}\boldsymbol{do}(\text{Person}, \text{ES}, \text{SeeResults})$$
$$cx_6 \rightarrow \texttt{PE}_{\text{Governor1}}\boldsymbol{do}(\text{NonResident}, \text{PES}_{c_1}, \text{SeeResults})$$

In the above policy rules, Governor1 prevents NonResident people to see the results of presidential election in city 1 for 24 hours after the elections. It is obvious that Resident people can see the results following the Interior Minister's statement.

A.2 Distributed Semantic Digital Library

In Sharif University of Technology, each department had a set of digital scientific resources in its local digital library. To make all the scientific resources accessible to different users and having inter-university collaboration for science evolution, the deans of the departments decided to share their resources with the central library of the university. The central library has a set of resources other than the resources shared by the departments, and also has access to the resources of some scientific institutes (e.g., IET, IEEE, ACM, and ScienceDirect). Figure A.3 shows some of the security domains of this distributed digital library. In this case we take two departments (i.e., Dept. of Comp. Eng. and Dept. of Math. Sci.) and one scientific institute (i.e., IEEE).

Each department has its security policy rules for its shared scientific resources other than the policies specified by the central library. Each resource itself may has some policies regrading the author's or publisher's copyrights (good examples of such policies can be found in `http://books.goolge.com`). Some of the departments (e.g., Dept. of Math. Sci.) delegate the security management of the shared resources to the central library. Thus, central library specifies the security policies on behalf them to access the shared resources.

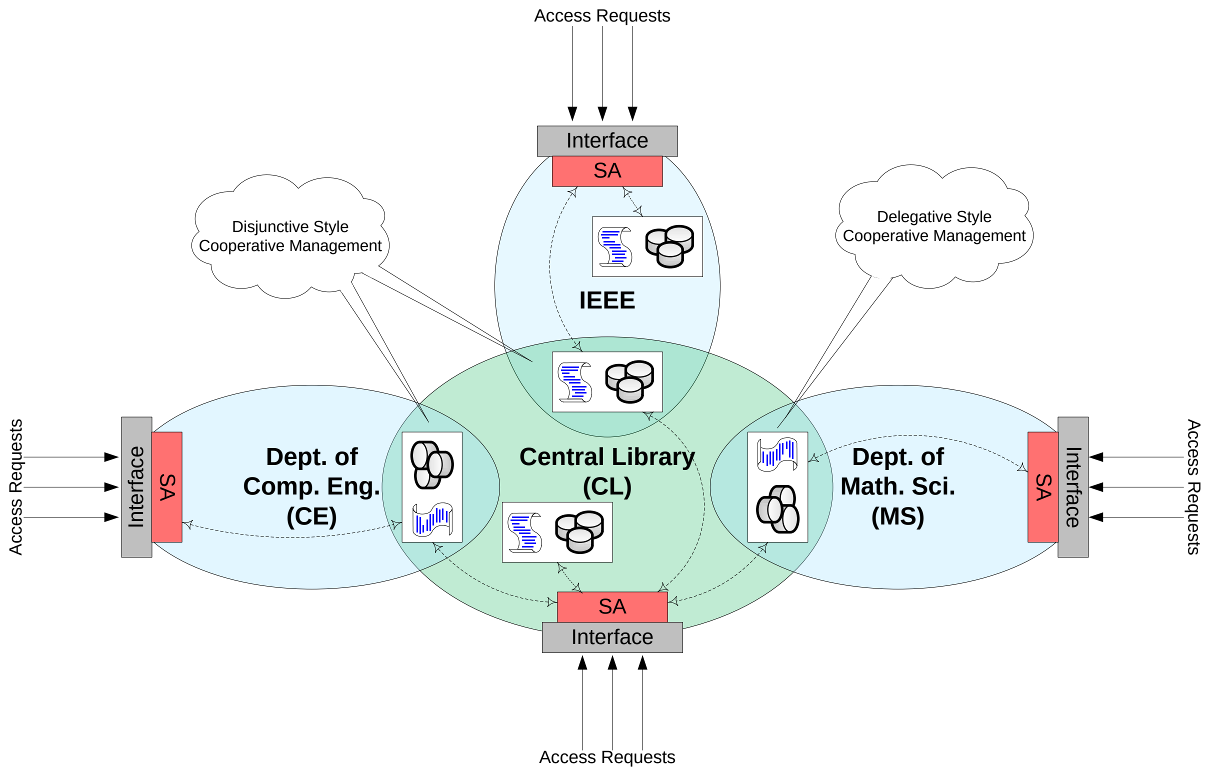


Figure A.3: Security domains in distributed semantic digital library.

However some of the others (e.g., Dept. of Comp. Eng.) cooperatively with the central library manage their shared digital resources. The shared subdomain between each foreign scientific institute and the central library is cooperatively managed based on the disjunctive management style. This means, everybody can access these resources if the rules of the institute or the rules of the central library allow.

A.2.1 Semantic Technology and Security Management

Due to the high volume of digital scientific resources and different types of the users access to them, semantic technology is used to describe the resources, the users, and the action types to access the resources. Figures A.4, A.5, and A.6 show parts of the ontologies of the users (subjects), the resources (objects), and the actions respectively. Note that the ontologies shown in these figures are not complete and many details (e.g., the attributes that each concept has and many other concepts) are eliminated to decrease the complexity of the case

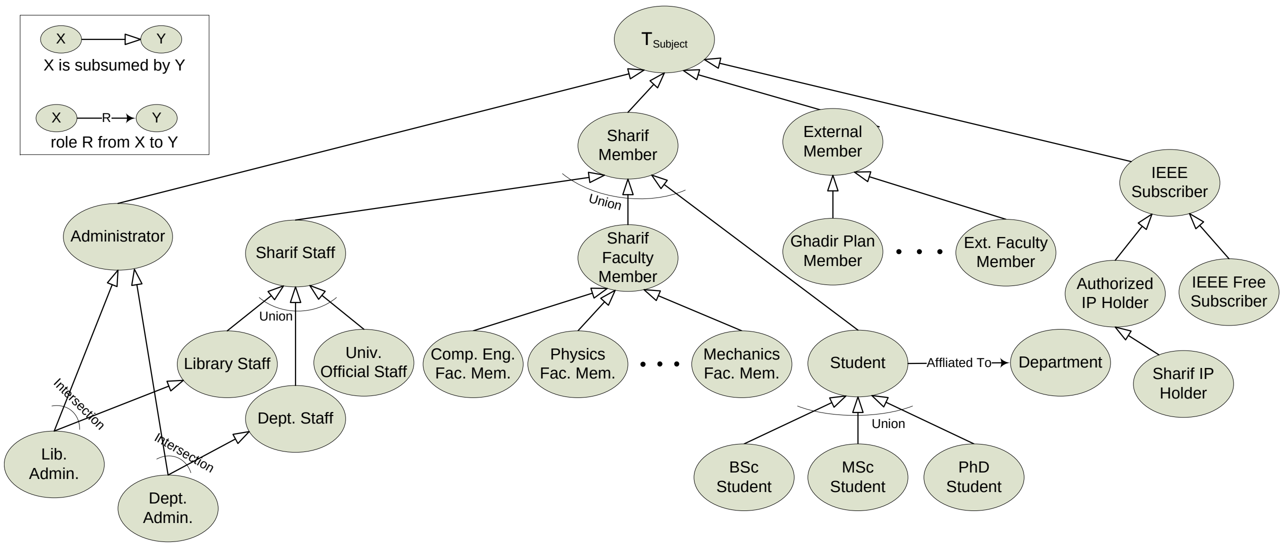


Figure A.4: Subjects ontology in the distributed semantic digital library case study.

study.

Following the formal specification of the authorization model in Section 6.1, it is clear that the definition of the elements of FDS is not difficult in this case study (e.g., $\hat{\mathcal{O}}$ is the set of all resources in the distributed digital library). Note that the authority of each security domain is denoted by $auth_X$, where X is the abbreviation of the domain's name. The shared domain between domains X and Y is also denoted by $X \between Y$.

For each security domain X in this case study, SKB $\mathcal{K}_X$ is defined as $\mathcal{K}_X = \langle \mathcal{TB}_X, \mathcal{AB}_X, \mathcal{SB}_X \rangle$, where $\mathcal{TB}_X$ contains the descriptions of the ontologies shown graphically in figures A.4, A.5, and A.6.

The assertions introducing the resources registered in the domain X are stored in $\mathcal{AB}_X$. For example for a conference paper $p1$, we have the assertion $ConferencePaper(p1)$ in $\mathcal{AB}_X$. For each action concept, we define an instance with the same name (e.g., *edit* for the action concept *Edit*). Thus, the assertions about the actions are defined as follows.

$$Edit(edit),\ Upload(upload),\ Delete(delete),\ Read(read),\ Download(download),$$
$$OnlineRead(onlineread),\ OnlinePreview(onlinepreview),\ Search(search),\ Print(print)$$

Note that in this way, for example *read* is an instance of *Read* and also is an instance of *Edit*, which means *read* is a kind of edit action; where we read, but do not write.

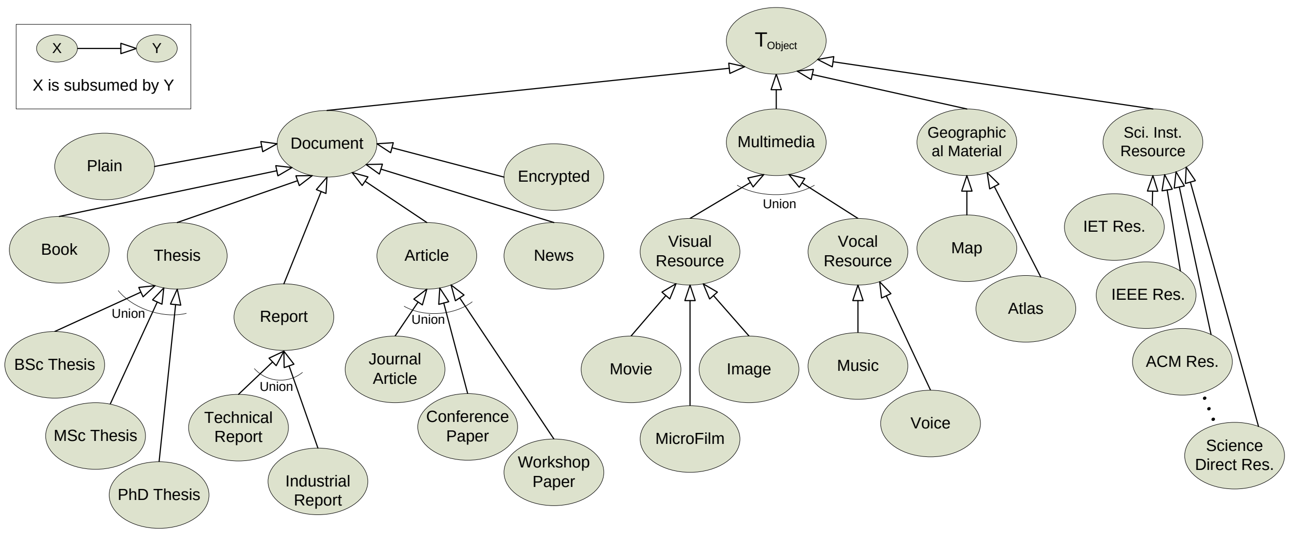


Figure A.5: Objects (resources) ontology in the distributed semantic digital library case study.

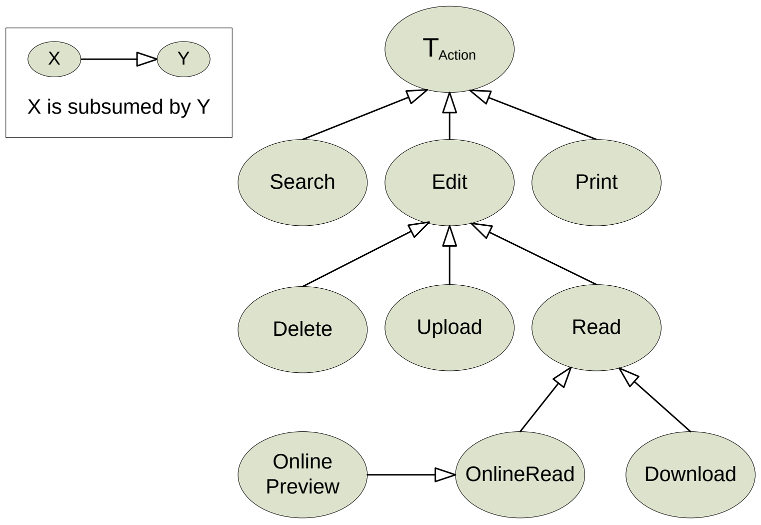


Figure A.6: Actions ontology in the distributed semantic digital library case study.

The assertions about the requesters (subject instances) are inserted dynamically during the access control procedure as described in Section 6.3.3.

A.2.2 Security Policy Rules

The samples of concept-level security policy rules specified in each security domain in this case study (stored in $\mathcal{SB}_X$) are as follows.

Dept. of Comp. Eng. (CE)

The policy rules specified in domain CE and its shared subdomains are as follows.

- The students of CE can have read access to the CE's resources.
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CE}@CE)} do(Student \sqcap \exists AffliatedTo.\{compEng\}, \top_{Object}, Read)$

- Visiting professors and students in CE can access to CE's resources except theses.
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CE}@CE)} do((Student \sqcup ExtFacultyMember) \sqcap Visitor,$
  $\top_{Object} \setminus Thesis, Read)$

- The shared domain $CE \between CL$ is defined as a subdomain of CE, i.e., $CE \between CL \preceq CE$.

Central Library (CL)

The policy rules specified in domain CL and its related shared subdomains are as follows.

- Graduate level students (i.e., MSc and PhD students) and Sharif faculty members can get read access to the resources of external scientific institutes including IEEE.
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CL}@IEEE \between CL)} do(MScStudent \sqcup PhDStudent \sqcup SharifFacultyMember,$
  $SciInstResource, Read)$

- The members of Ghadir plan and faculty members of other universities can only read the CE's theses online.
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CL}@CE \between CL)} do(GhadirPlanMember \sqcup ExtFacultyMember, Thesis,$

$OnlineRead)$

- PhD students of Sharif can upload their theses and technical reports to the digital library; however, MSc and BSc theses should be added by the library's administrator.
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CL}@CL)} do(PhDStudent, Thesis \sqcup TechnicalReport, Upload)$
  $\mathtt{T} \rightarrow \mathtt{IM}_{(auth_{CL}@CL)} do(Student \setminus PhDStudent, Thesis, Upload)$
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CL}@CL)} do(LibAdmin, Thesis, Edit)$

- Authority of domain Dept. of Math. Sci. (MS) should grant read right to all Sharif members as well as the external members to access all registered resources in domain MS. Similarly, full access to the administrators of central library.
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CL}@MS)}(\mathtt{PE}_{(auth_{MS}@MS)} do(SharifMember \sqcup ExternalMember,$
  $\top_{Object}, Read))$
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CL}@MS)}(\mathtt{PE}_{(auth_{MS}@MS)} do(LibAdmin, \top_{Object}, \top_{Action}))$

IEEE Institute

The policy rules specified in domain IEEE and its shared subdomains are as follows.

- The users connected with Sharif's IP can get read access to the shared resources with Sharif during year 2010 (that they have contract).
  $YearIs2010 \rightarrow \mathtt{PE}_{(auth_{IEEE}@IEEE \between CL)} do(SharifIPholder, \top_{Object}, Read)$

- The free subscribed users of IEEE digital library (with username and password) freely have read access to the papers of conferences and workshops in the case that the traffic is less than 120 M.
  $TrafficLessThan120M \rightarrow \mathtt{PE}_{(auth_{IEEE}@IEEE)} do(IEEEsubscriber,$
  $ConferencePaper \sqcup WorkshopPaper, Read)$

- The shared domain $IEEE \between CL$ is defined as a subdomain of $IEEE$, i.e., $IEEE \between CL \preceq IEEE$.

Ground-Level Policy Rules

The samples of ground-level policy rules in this case study are as follows.

- The book entitles 'Computer Security Basics' with ID 'qa76.9' cannot be downloaded by non-library-administrators.
  $\mathtt{T} \rightarrow \mathtt{IM}_{(auth_{CE}@CE\between CL)} do(\top_{Subject} \setminus LibAdmin, N_{qa76.9}, Download)$

- The content of the above book (with ID 'qa76.9') can be searched by all users and if they require the results of the search can be encrypted.
  $\mathtt{T} \rightarrow \mathtt{PE}_{(auth_{CE}@CE\between CL)} do(\top_{Subject}, N_{\mathrm{qa76.9}}, Search)$
  $\wedge\ \mathtt{OB}_{(auth_{CE}@CE\between CL)} cap(Encrypted, N_{\mathrm{qa76.9}})$

- The users who are not the faculty members of Sharif Univ. cannot see the aerial map with ID 'm103'.
  $\mathtt{T} \rightarrow \mathtt{IM}_{(auth_{CL}@CL)} do(\top_{Subject} \setminus SharifFacultyMember, N_{\mathrm{m103}}, OnlinePreview)$

Appendix B

MA(DL)^2 Reasoner in Prolog

The source code of the MA(DL)^2 reasoner in Prolog is presented in this appendix. The following code has been run and tested in SWI-Prolog version 5.7.15. To use the reasoner, TBox $\mathcal{TB}$ and SBox $\mathcal{SB}$ of the MA(DL)^2 security knowledge base should be filled using the following commands.

```
    assert(tbox([comma separated list of t-formulae])).
    assert(sbox([comma separated list of c-formulae])).
```

For checking the validity (or proof) of a given c-formula F, `proof(F)` should be called.

```
/************************************************************************
Theorem prover of MA(DL)^2 logic using an analytic tableaux algorithm


Version:  2.07
Language: SWI Prolog
Date:     2010.07.31
Author:   Morteza Amini
Copyright (C) 2008-2010 Morteza Amini
************************************************************************/
:- op(300,xfy,#),   % disjunctive authority
   op(310,xfy,&),   % cooperative authority
   op(320,xfy,>),   % delegative authority
   op(360,xfy,<),   % subsumption
   op(400,fy,-),    % negation
   op(500,xfy,^),   % conjunction
   op(600,xfy,v),   % disjunction
   op(650,xfy,=>),  % implication
   op(700,xfy,<=>), % equivalence
   op(800,xfy,@).   %authority at domain

/***********************************************************************
Negation Normal Form (nnf)
Usage: nnf(+Fml,-NNF)
***********************************************************************/
nnf(ob(U,F),NNF) :- !, nnf(F,NNF1), cnf(NNF1,CNF), obRule(ob(U,CNF), NNF).
nnf(pe(U,F),NNF) :- !, nnf(F,NNF1), dnf(NNF1,DNF), peRule(pe(U,DNF), NNF).
nnf(pred(P,X),pred(P,Y)) :- !, expand_param(X,Y).
nnf(-pred(P,X),-pred(P,Y)) :- !, expand_param(X,Y).
nnf(A ^ B,NNF) :- !, nnf(A,NNF1), nnf(B,NNF2), NNF = (NNF1 ^ NNF2).
nnf(A v B,NNF) :- !, nnf(A,NNF1), nnf(B,NNF2), NNF = (NNF1 v NNF2).
nnf(F,NNF) :-
    (F = -(-A)      -> F1 = A;
     F = -ob(U,A)   -> F1 = pe(U,-A);
     F = -pe(U,A)   -> F1 = ob(U,-A);
     F = im(U,A)    -> F1 = ob(U,-A);
     F = -im(U,A)   -> F1 = pe(U,A);
     F = gr(U,A)    -> F1 = pe(U,-A);
     F = -gr(U,A)   -> F1 = ob(U,A);
     F = -(A v B)   -> F1 = (-A ^ -B);
     F = -(A ^ B)   -> F1 = (-A v -B);
     F = (A => B)   -> F1 = (-A v B);
     F = -(A => B)  -> F1 = (A ^ -B);
     F = (A <=> B)  -> F1 = ((A ^ B) v (-A ^ -B));
     F = -(A <=> B) -> F1 = ((A ^ -B) v (-A ^ B)))
    ,!,nnf(F1,NNF).
nnf(P,P).
```

```
obRule(ob(U,(A ^ B)), (OBA ^ OBB)) :- !, obRule(ob(U,A),OBA), obRule(ob(U,B),OBB).
obRule(F, F).
peRule(pe(U,(A v B)), (PEA v PEB)) :- !, peRule(pe(U,A),PEA), peRule(pe(U,B),PEB).
peRule(F, F).

/**********************************************************************
Expansion of Parameteres of Concept Predicates
Usage: expand_param(+X,-EX)
**********************************************************************/
expand_param([],[]):-!.
expand_param([NC|X],[EC|Y]) :- expand_defs(NC,D),
            negnormform(D,E), disnormform(E,EC), !, expand_param(X,Y).

/**********************************************************************
Disjunction Normal Form of a Concept Description
Usage: disnormform(+NNF,-CNF)
**********************************************************************/
disnormform(or(A, B), or(A1, B1)):- !, disnormform(A, A1),
disnormform(B, B1).
disnormform(and(A, B),DNF):- !, disnormform(A, A1),
    disnormform(B, B1), disnormform1(and(A1, B1), DNF).
disnormform(DNF,DNF).

disnormform1(and(A, or(B, C)), or(A1, B1)):- !,
disnormform1(and(A, B), A1), disnormform1(and(A, C), B1).
disnormform1(and(or(A, B), C), or(A1, B1)):- !,
disnormform1(and(A, C), A1), disnormform1(and(B, C), B1).
disnormform1(DNF,DNF).

/**********************************************************************
Conjunctive Normal Form (cnf)
Usage: cnf(+NNF,-CNF)
**********************************************************************/
cnf(A ^ B, (A1 ^ B1)):- !, cnf(A, A1), cnf(B, B1).
cnf(A v B,CNF):- !, cnf(A, A1), cnf(B, B1), cnf1(A1 v B1, CNF).
cnf(CNF,CNF).

cnf1((A ^ B) v C, (A1 ^ B1)):- !, cnf1(A v C, A1), cnf1(B v C, B1).
cnf1(A v (B ^ C), (A1 ^ B1)):- !, cnf1(A v B, A1), cnf1(A v C, B1).
cnf1(CNF,CNF).

/*********************************************************************
Disjunctive Normal Form (dnf)
Usage: nnf(+NNF,-DNF)
*********************************************************************/
dnf(A v B, (A1 v B1)):- !, dnf(A, A1), dnf(B, B1).
dnf(A ^ B,DNF):- !, dnf(A, A1), dnf(B, B1), dnf1(A1 ^ B1, DNF).
dnf(DNF,DNF).

dnf1(A ^ (B v C), (A1 v B1)):- !, dnf1(A ^ B, A1), dnf1(A ^ C, B1).
dnf1((A v B) ^ C, (A1 v B1)):- !, dnf1(A ^ C, A1), dnf1(B ^ C, B1).
dnf1(DNF,DNF).

/*********************************************************************
Usage: proof(+F) --> checks the proof of an MA(DL)^2 formula F.
*********************************************************************/

:- dynamic(sbox/1).
:- dynamic(tbox/1).

proof(F) :- ( (tbox(TB) -> buildOnt(TB)); \+tbox(_)),
          ( (sbox(SB) -> (buildFormula(SB,SBF), S=SBF ^ -F)) ;
(\+ sbox(_) -> S= -F) ),
    unsat(S),    %run tableaux algorithm
    retractall(ont(_)). %remove assertions from SKB.
proof(_) :- retractall(ont(_)), fail.

unsat(F) :- nnf(F,NNF), dnf(NNF,DNF), !, tab([DNF], [], [*], []).
```

```
buildFormula([X],X):- !.
buildFormula([A|SB],SBF ^ A) :- buildFormula(SB,SBF).

buildOnt([]):-!.
buildOnt([equiv(X,X)|TB])  :- !, buildOnt(TB). %ignore self loop
buildOnt([subsum(X,X)|TB]) :- !, buildOnt(TB). %ignore self loop
buildOnt([equiv(X,Y)|TB])  :- !, assert(ont(equiv(X,Y))), buildOnt(TB).
buildOnt([subsum(X,Y)|TB]) :- !, concat(X,'__restrict__temp',Z),
assert(ont(equiv(X,and(Y,Z)))), buildOnt(TB).

/**************************************************************************
Anlytical Tableaux Algorithm
Usage: tab(+F, +L, +OBF, +PEF)
**************************************************************************/

%################ FRAGMENTATION AND CLASSIFICATION RULES ################
tab([A ^ B|F], L, OB, PE)  :- !, tab([A,B|F], L, OB, PE).  % Alpha Rule
tab([A v B|F], L, OB, PE)  :- !, tab([A|F], L, OB, PE),
   tab([B|F], L, OB, PE).  % Beta Rule

tab([ob((U1#U2)@D,X)|F], L, OB, PE) :- !, tab([ob(U1@D,X) ^ ob(U2@D,X)|F], L, OB, PE).
tab([pe((U1#U2)@D,X)|F], L, OB, PE) :- !, tab([pe(U1@D,X) v pe(U2@D,X)|F], L, OB, PE).

tab([ob((U1>U2)@D,X)|F], L, OB, PE) :- !, tab([ob(U1@D,ob(U2@D,X))|F], L, OB, PE).
tab([pe((U1>U2)@D,X)|F], L, OB, PE) :- !, tab([pe(U1@D,pe(U2@D,X))|F], L, OB, PE).

tab([pred(P,X)|F],  L, OB, PE)  :- !,
   ( (hasUnion(X) -> (unionDecom(X,X1,X2),
  tab([pred(P,X1),pred(P,X2)|F], L, OB, PE))); % Lambda-{+U} Rule
     tab(F, [pred(P,X)|L], OB, PE) ).  % Gamma-L Rule

tab([-pred(P,X)|F],  L, OB, PE)  :- !,
   ( (hasUnion(X) -> (unionDecom(X,X1,X2),
  tab([-pred(P,X1) v -pred(P,X2)|F], L, OB, PE))); % Lambda-{-U} Rule
     tab(F, [-pred(P,X)|L], OB, PE) ).  % Gamma-L Rule

tab([ob(U@D,A)|F], L, OB, PE):- !, tab(F, L, [ob(U@D,A)|OB], PE). % Gamma-O
tab([pe(U@D,A)|F], L, OB, PE):- !, tab(F, L, OB, [pe(U@D,A)|PE]). % Gamma-P
tab([subdomain(D1,D2)|F],  L, OB, PE)  :- !,
tab(F, [subdomain(D1,D2)|L], OB, PE).  % Gamma-L Rule
tab([-subdomain(D1,D2)|F],  L, OB, PE) :- !,
tab(F, [-subdomain(D1,D2)|L], OB, PE). % Gamma-L Rule
tab([P|F], L, OB, PE)  :- !, tab(F, [P|L], OB, PE).% Gamma-L Rule

%################### SUB-TABLEAUX CONSTRUCTION RULES ###################
tab([], L, OB, [pe(U@D,X)|PE]) :- removeOB(L, OB, U, D, FOB), !,
(tab([X|FOB], [], [*], []); tab([], L, OB, PE)). % Kappa Rule
tab([], L, [ob(U@D,X)|OB], []) :- removeOB(L, OB, U, D, FOB), !,
  (tab([X|FOB], [], [*], []); tab([], L, [OB|ob(U@D,X)], [])). % Theta Rule
tab([], L, _, _) :- !, contradiction(L).

%############################# CLOSURE RULES #############################
contradiction(L) :- (member(false, L); member(-true, L)), !.
contradiction(L) :- member(P,L),  member(-P,L), !.
contradiction(L) :- member(pred(P,X), L), member(-pred(P,Y), L),
subpred(Y, X), !.
contradiction(L) :- member(-subdomain(D1,D2),L), subdomain(L,D1,D2), !.

% return true if there is at least one union composition of
concepts in the arguments of a given predicate
hasUnion([or(_,_)|_]) :- !.
hasUnion([_|X]) :- !, hasUnion(X).

% takes a list of concepts and decomposes the first union composition
(e.g., decomposes or(C1,C2) to C1 and C2) and separates it to two lists.
unionDecom([], [], []).
unionDecom([or(C1,C2)|X], [C1|Y1], [C2|Y2]) :- !, copy(X,Y1), copy(X,Y2).
unionDecom([C|X],[C|Y1],[C|Y2]) :- !, unionDecom(X,Y1,Y2).

% copies a list into another
copy([],[]).
copy([A|X],[A|Y]) :- copy(X,Y).

% removes ob(u@d1,x) from OB where d1 is subdomain of d, and collects x's in FOB.
removeOB(_,[], _, _, []) :- !.
removeOB(L,[ob(U@D1,X)|OB], U, D, [F|FOB])  :- subdomain(L, D, D1), F=X, !,
                                                removeOB(L,OB, U, D, FOB).
removeOB(L,[_|OB], U, D, FOB) :- !, removeOB(L, OB, U, D, FOB).

%checks whether p(x) is sub-predicate of p(x') based on subsumption inference
service (must be obtained by an appropriate inference engine from TBox (TB)).
subpred([], []) :- !.
subpred([HX|X], [HY|Y]) :- (HX=HY ; subsume(HX, HY)), !, subpred(X, Y).

subsume(C1,C2):- tab_proof(subsum(C1,C2)).

subdomain(L, D1, D2)  :- (D1=D2 ; subdomain1(L, D1, D2)), !.
subdomain1(L, D1, D2) :- member(subdomain(D1, D2), L).
subdomain1(L, D1, D2) :- member(subdomain(D1, D3), L),
subdomain1(L, D3, D2).

%############################## ALC REASONER  ##############################
% Tableaux reasoner for ALC description logic, by Thomas Herchenroder,
University of Edinburgh, 2006.
:- use_module(library(lists)).
:- op(100,fy,~).
:- dynamic ont/1.
:- dynamic id/1.
% Main Proof Goal
tab_proof(Exp) :- % true/false = satisfiable/unsatisfiable
alc_proof(Exp).
```

```
%construct_goal
alc_proof(equiv(A,B)) :- \+ tabl(and(A,~B)), \+ tabl(and(B,~A)).
alc_proof(subsum(A,B)) :- \+ tabl(and(A,~B)).
alc_proof(disjoint(A,B)) :- \+ tabl(and(A,B)).
alc_proof(unsat(A)) :- % unsatisfiable A
            \+ tabl(A).
alc_proof(A) :- % try to satisfy everything else
            tabl(A).

% main worker
tabl(Exp) :- expand_defs(Exp,Exp1), % expand expression into most basic
             negnormform(Exp1,Exp2), % NNF transformation
             setID(0), !,
             search([[Exp2]],df,_). % do the proof as an agenda search

%negation normal form
negnormform(~ ~X,X1) :- negnormform(X,X1).
                        negnormform(~forall(R,C),exist(R,C1)) :-
                        negnormform(~C,C1).
negnormform(forall(R,C),forall(R,C1)) :- negnormform(C,C1).
negnormform(~exist(R,C),forall(R,C1)) :- negnormform(~C,C1).
negnormform(exist(R,C),exist(R,C1)) :- negnormform(C,C1).
negnormform(~and(A,B),or(A1,B1)) :- negnormform(~A,A1),
                                    negnormform(~B,B1).
negnormform(and(A,B),and(A1,B1)) :- negnormform(A,A1),
                                    negnormform(B,B1).
negnormform(~or(A,B),and(A1,B1)) :- negnormform(~A,A1),
                                    negnormform(~B,B1).
negnormform(or(A,B),or(A1,B1)) :- negnormform(A,A1),
                                  negnormform(B,B1).
negnormform(~X,~X) :- atom(X).
negnormform(X,X) :- atom(X).

%expansion of definitions
```

```
expand_defs(forall(R,C),forall(R,C1)) :- expand_defs(C,C1).
expand_defs(exist(R,C),exist(R,C1)) :- expand_defs(C,C1).
expand_defs(and(A,B),and(A1,B1)) :- expand_defs(A,A1), expand_defs(B,B1).
expand_defs(or(A,B),or(A1,B1)) :- expand_defs(A,A1), expand_defs(B,B1).
expand_defs(~A,~A1) :- expand_defs(A,A1).
expand_defs(X,Y) :- atom(X), ont(equiv(X,X1)), expand_defs(X1,Y).
expand_defs(X,X) :- atom(X), \+ ont(equiv(X,_)).

% search(+Goal,+Style,-ResultList) -- agenda style search
% -- transforms Goal into a list of [clash]/[model] elements
search([],_,[]).
search(Reduced, _, Reduced) :- Reduced = [H|_],
                               H = [clash], % a clash leaf fails the proof
                               !, fail.
search([Node| T], Style, Reduced) :-
                                process_node(Node,NewNodes),
                                filter_nodes(NewNodes,NewNodes1),
                                merge_agendas(NewNodes1, T, Style, New),
                                search(New, Style, Reduced).
filter_nodes(NewNodes,NewNodes1) :-
( setof(X,(member(X,NewNodes),X\=[model]),NewNodes1);
                                NewNodes1 = []), !.
merge_agendas(A1, A2, df, New) :- append(A1, A2, New),!.
merge_agendas(A1, A2, bf, New) :- append(A2, A1, New),!.

% reduce a node of the proof tree
process_node([clash],[]) :- !.
process_node([model],[]) :- !.
process_node(ListOfDLExps,ResultListOfLists) :-
                                    transform_connect(ListOfDLExps,_,LoL3),
                                    expand_nodes(LoL3,ResultListOfLists).
expand_nodes([],[]).
expand_nodes([H|R],RLoL) :-
                        expand_node(H,RLoL1),
                        expand_nodes(R,RLoL2),
                        append(RLoL1,RLoL2,RLoL).

% process a single node
expand_node([],[]).
expand_node([model],[[model]]) :-!.
expand_node([clash],[[clash]]) :-!.
expand_node(Node,[N1]) :- check_clash(Node,N1),!. % clash closes this branch
expand_node(Node,N1) :- expand_exist(Node,[],LoE), % expand existential restrictions
                        LoE \= [], % only continue with non-empty edges
                        expand_forall(Node,LoE,LoE2), % try eliminate value
                                                        restrictions
                        extract_nodes(LoE2,N1). % get the list of new fringe nodes
expand_node(Node,[N1]) :- % model closes this branch
                        expand_exist(Node,[],LoE),
                        LoE = [],N1 = [model].

% extract list of nodes from list of edges
extract_nodes([],[]).
extract_nodes([edge(_,_,N)|R],[N|R1]) :- extract_nodes(R,R1).

% transform and/or connectives
transform_connect([],_,[[]]).
transform_connect([and(A,B)|R],N,R1) :-
                        ( member(A,R) ->
                        ( member(B,R) ->
                        transform_connect(R,N,R1);
                        transform_connect([B|R],N,R1));
                        ( member(B,R) ->
                        transform_connect([A|R],N,R1);
                        transform_connect([A,B|R],N,R1))).
transform_connect([or(A,B)|R],N,LoL) :-
                        ( \+ (member(A,R) ; member(B,R)) ->
                        ( transform_connect([A|R],N,LoL);
                        transform_connect([B|R],N,LoL));
                        transform_connect(R,N,LoL)).
transform_connect([H|R],N,LoL) :-
                        H \= and(_,_),
                        H \= or(_,_),
                        transform_connect(R,N,LL1),
                        LL1 = [LL2],
                        LoL = [[H|LL2]].

% transform forall/exist quantifiers
expand_exist([],L,L).
expand_exist([exist(R,C)|T],T1,L) :-
            ( (member(edge(R,_,X),T1), member(C,X) ) -> % existing R edge
              expand_exist(T,T1,L); getID(Id),
              expand_exist(T,[edge(R,Id,[C])|T1],L) ).
expand_exist([H|T],T1,L) :- H \= exist(_,_), expand_exist(T,T1,L).
expand_forall(_,[],[]).

% push concept into exist. node
expand_forall(Node,[edge(R,I,N)|RoE],[edge(R,I,N2)|R1]) :-
                                setof(X,member(forall(R,X), Node),C1),
                                append(C1,N,N1),
                                remove_duplicates(N1,N2),
                                expand_forall(Node,RoE,R1).
expand_forall(Node,[edge(R,I,N)|RoE],[edge(R,I,N)|LoE]) :-
                                \+ member(forall(R,_),Node),
                                expand_forall(Node,RoE,LoE).
check_clash(Exp,Exp1) :- member(A,Exp), member(~A,Exp), Exp1 = [clash].
remove_duplicates([],[]).
remove_duplicates([H|T],[H|R]) :- r_d(H,T,[],R1),
remove_duplicates(R1,R), !.
r_d(_,[],T,T).
r_d(E,[E|T],TT,T1) :- r_d(E,T,TT,T1).
r_d(E,[X|T],TT,T1) :- X \= E, r_d(E,T,[X|TT],T1).
```

```
getID(I):- id(I1), I is I1 + 1, retract(id(I1)), asserta(id(I)), !.
getID(I):- \+ id(_), I is 0, asserta(id(I)),!.
setID(X):- ( id(Y) -> retract(id(Y)); true), asserta(id(X)).
```

Appendix C

Extended Mindswap Test Cases

The test cases generated based on the Extended Mindswap test cases are presented in this appendix. In each test case, TBox $\mathcal{TB}$ is filled with `assert(tbox([...]))` and SBox $\mathcal{SB}$ is filled with `assert(sbox([...]))`. `madl2_query()` represents the query on the MA(DL)^2 inference engine, and `alc_query()` represents the query on the $\mathcal{ALC}$ inference engine.

```
1- ex1_10
====================
assert(tbox([equiv(veggiepizza,and(pizza, forall(hastopping,(~meat)))), equiv(meatpizza,
and( pizza, forall( hastopping, (~veggie)))),equiv(veggie,or( mushroom, olive)),equiv
(meat,or( pepperoni ,sausage))])).
assert(sbox([ob(u@d,pred(p,[veggiepizza])), ob(u@d, pred(p,[meatpizza]))])).
madl2_query(pe(u@d, pred(p,[and(veggiepizza,meatpizza)]))).
alc_query(and(veggiepizza, meatpizza))).

2- ex1_11
====================
assert(tbox([equiv(a,and( h, and( i, (~d)))), equiv(j,( ~k)), equiv(b,( ~g)), equiv
(d,forall( q, j)), equiv(g,( ~e))])).
assert(sbox([ob(u@d, pred(p,[exist(p,a)])), ob(u@d, pred(p,[exist(p,b)])), ob(u@d,
pred(p,[c])), ob(u@d, pred(p,[d])), ob(u@d, pred(p,[~exist(p,(~and((~e),f)))])]))).
madl2_query(pe(u@d,pred(p,[and( exist( p, a), and(exist( p, b), and(and( c, d),
(~exist( p, (~and((~e), f)))))))]))).
alc_query(and( exist( p, a), and(exist( p, b), and(and( c, d), (~exist( p, (~and((~e),
f))))))))).
```

```
3- ex1_1
====================
assert(tbox([equiv(beer,and( drink,and(exist( hasingr, water),and( exist( hasingr,hops),
and( exist( hasingr, malt), forall( hasingr, or( water, or(hops, malt)))))))), equiv(
grapes,and( (~hops),and( (~malt), (~water)))), equiv(wine,and( drink, exist( hasingr,
grapes)))])).
assert(sbox([ob(u@d,pred(p,[wine])), ob(u@d,pred(p,[beer]))])).
madl2_query(pe(u@d,pred(p,[and( wine, beer)]))).
alc_query(and( wine, beer)).

4- ex1_2_1
====================
assert(sbox([ob(u@d,pred(p,[exist( r, b)])), ob(u@d, pred(p,[forall( r, (~b))]))])).
madl2_query(pe(u@d,pred(p,[and( exist( r, b), forall( r, (~b)))]))).
alc_query(and( exist( r, b), forall( r, (~b)))).

5- ex1_2_2
====================
assert(sbox([ob(u@d,pred(p,[exist( r, b)])), ob(u@d,pred(p,[forall( r, or( a, (~b)))]))])).
madl2_query(pe(u@d,pred(p,[and( exist( r, b), forall( r, or( a, (~b))))]))).
alc_query(and( exist( r, b), forall( r, or( a, (~b))))).

6- ex1_2_3
====================
assert(tbox([equiv(aa,or( a, (~a))), equiv(a,exist( r, exist( r, exist( r, c)))),
equiv(b,and( a, and(exist( r, aa), or( aa, (~aa)))))])).
assert(sbox([ob(u@d,pred(p,[exist( r, b)])), ob(u@d, pred(p, [forall( r, (~b))]))])).
madl2_query(pe(u@d,pred(p,[and( exist( r, b), forall( r, (~b)))]))).
alc_query(and( exist( r, b), forall( r, (~b)))).

7- ex1_2_4
====================
assert(tbox([equiv(a,(~b))])).
assert(sbox([ob(u@d,pred(p,[exist( r, b)])), ob(u@d, pred(p, [forall( r, or( a, (~b)))]))])).
madl2_query(pe(u@d,pred(p,[and( exist( r, b), forall( r, or( a, (~b))))]))).
alc_query(and( exist( r, b), forall( r, or( a, (~b))))).

8- ex1_3
====================
assert(tbox([equiv(werewolf,and(animal, and(exist(haspower, magical),forall(speaks,
language)))),equiv(acramantula,and( beast,and(or( male,female),exist( haspower,magical
)))), equiv(wizard,and(male,and(human,exist( haspower, magical)))), equiv(centaur,and(
animal,and((~human), and(or( male,female),and(exist( haspower, magical),forall( speaks,
language)))))), equiv(vampire,and( beast,and(or( male, female),forall( haspower,
magical)))), equiv(beast,and(animal,(~human))),equiv(muggle,and( human,and( or( male,
female),forall( haspower,(~magical))))),equiv(witch,and(female,and(human, exist(haspower,
magical)))), equiv(human,and( animal,exist(speaks, language))),equiv(male,and( animal,
(~female))), equiv(merpeople,and( animal,and((~human),and(forall( haspower,magical),
forall(speaks, language)))))])).
assert(sbox([ob(u@d,pred(p,[werewolf])), ob(u@d, pred(p, [human]))])).
madl2_query(pe(u@d,pred(p,[and( werewolf, human)]))).
alc_query(and( werewolf, human)).
```

9- ex1_4

```
====================
assert(tbox([equiv(salto,and( cartwheel, and( (~exist( hasposition, handsonfloor)),
exist( hasposition, twist)))), equiv(fronttuck,and((~cartwheel),forall( hasposition,
tuck))), equiv(roundoff,and( cartwheel,and( handstand ,forall( hasposition,pike)))),
equiv(backwalkover,and( exist( hasposition, bridge),exist( hasposition, handstand))),
equiv(forwardroll,or( exist( hasposition, pike),or( exist(hasposition,straddle), exist(
hasposition, tuck)))), equiv(backhandspring,exist( hasposition,bridge)), equiv(handstand,
(~forall( hasposition, pike)))])).
assert(sbox([ob(u@d,pred(p,[roundoff])), ob(u@d, pred(p,[~backhandspring]))])).
madl2_query(pe(u@d,pred(p,[and( roundoff, (~backhandspring))]))).
alc_query(and( roundoff, (~backhandspring))).
```

10- ex1_5

```
====================
assert(tbox([equiv(werewolf,and( beast,and( exist( haspower, magical), forall( speaks,
language)))), equiv(acramantula,and( beast ,and(or( male,female), exist( haspower,
agical)))),equiv(wizard,and( male, and( human, exist( haspower, magical)))), equiv(
centaur,and(animal,and((~human),and(or( male, female), and( exist(haspower, magical),
forall( speaks,language)))))), equiv(vampire,and( beast,and( or( male,female), forall(
haspower,magical)))),equiv(beast,and( animal, (~human))), equiv(muggle,and( human,and(
or( male, female), forall(haspower, (~magical))))), equiv(witch,and( female,and( human,
exist( haspower, magical)))),equiv(human,and( animal, exist( speaks, language))), equiv(
male,and( animal, (~female))),equiv(merpeople,and( animal,and( (~human),and( forall(
haspower, magical), forall( speaks,language)))))])).
```

```
assert(sbox([ob(u@d,pred(p,[werewolf])), ob(u@d, pred(p,[human]))])).
madl2_query(pe(u@d,pred(p,[and(werewolf, human)]))).
alc_query(and( werewolf, human)).

11- ex1_6
====================
assert(tbox([equiv(academicfreelicense,and( freesoftwarelicense, exist( hasrestriction,
and( mustkeepdisclaimer, and((~mustdistributemods),and((~licenseisviral),
(~cannotredistribute))))))), equiv(commonpubliclicense,and( freesoftwarelicense,
exist(hasrestriction,and(mustkeepdisclaimer, and( mustdistributemods, and((~licenseisviral),
(~cannotredistribute))))))), equiv(publicdomainlicense,and( freesoftwarelicense ,exist(
hasrestriction,(~or(mustkeepdisclaimer, or( mustdistributemods, or(licenseisviral,
cannotredistribute))))))),equiv(lessergnupubliclicense,and( freesoftwarelicense, exist(
hasrestriction, and(mustkeepdisclaimer,and(mustdistributemods,and((~licenseisviral),
(~cannotredistribute))))))), equiv(commercialsoftwarelicense,exist( hasrestriction,
cannotredistribute)),equiv(softwarelicense,or( freesoftwarelicense,
commercialsoftwarelicense)),equiv(opensoftwarelicense,and(freesoftwarelicense, exist(
hasrestriction,and(mustkeepdisclaimer,and( mustdistributemods, and(licenseisviral,
(~cannotredistribute))))))),equiv(gnugeneralpubliclicense,and( freesoftwarelicense, exist(
hasrestriction, and(mustkeepdisclaimer, and(mustdistributemods,and(licenseisviral,
(~cannotredistribute)))))))])).
assert(sbox([ob(u@d,pred(p,[lessergnupubliclicense])), ob(u@d, pred(p,[forall(
hasrestriction,cannotredistribute)]))])).
madl2_query(pe(u@d,pred(p,[and( lessergnupubliclicense, forall(hasrestriction,
cannotredistribute))]))).
alc_query(and( lessergnupubliclicense, forall( hasrestriction,cannotredistribute))).

12- ex1_7
====================
assert(tbox([equiv(parentdog,or( papadog, mamadog)), equiv(husbanddog,and(maledog,
exist(haswife, femaledog))), equiv(maledog,and( dog, (~female))), equiv(papadog,
and(maledog ,exist( haschild, dog))), equiv(femaledog,and( dog, female)), equiv(wifedog,
and(femaledog, exist(hashusband, maledog))), equiv(puppy,and( or( maledog, femaledog),
and(exist( hasmother,mamadog),and( exist( hasfather, papadog) ,(~exist(haschild, dog)))))),
equiv(mamadog,and( femaledog, exist( haschild, dog)))])).
assert(sbox([ob(u@d,pred(p,[puppy])), ob(u@d, pred(p,[wifedog]))])).
madl2_query(pe(u@d,pred(p,[and( puppy, and(mamadog, wifedog))]))).
alc_query(and( puppy, and(mamadog, wifedog))).
```

```
13- ex1_8
====================
assert(sbox([ob(u@d,pred(p,[and( or( a, b), or( c, d))])), ob(u@d, pred(p, [and(
or( a, c), or( b, d))]))])).
madl2_query(pe(u@d,pred(p,[or( and( or( a, b), or( c, d)), and( or( a, c), or( b, d)))]))).
alc_query(or( and( or( a, b), or( c, d)), and( or( a, c), or( b, d)))).

14- ex1_9
====================
assert(tbox([equiv(slitheringdragon,and( dragon, forall( transportmode, (~or(flying,
walking))))), equiv(walkingdragon,and( dragon, exist(transportmode, walking))),
equiv(firedrake,and( drake, and(forall( elemental, fire), exist(disposition, foe)))),
equiv(icedrake,and( drake,and(forall( elemental, water), exist(disposition, foe)))),
equiv(orientaldragon,and( walkingdragon, and(exist(elemental, water),forall(disposition,
friend)))), equiv(drake,and( walkingdragon, and( exist( elemental, or( water, fire)),
forall( disposition, foe)))), equiv(hydra,and(or( slitheringdragon, flyingdragon),
exist( disposition,foe))), equiv(westerndragon,and( flyingdragon,and( forall( elemental,
or( earth,water)), exist(disposition, foe)))), equiv(wyrm,and( slitheringdragon,
exist( elemental, water))), equiv(flyingdragon,and( dragon, exist( transportmode,flying))),
equiv(dragonet,and( forall( disposition, foe),and( or( walkingdragon,flyingdragon), forall(
elemental, (~or( earth,water))))))])).
assert(sbox([ob(u@d,pred(p,[hydra])), ob(u@d, pred(p, [and(dragonet, exist( elemental,
fire))]))])).
madl2_query(pe(u@d,pred(p,[and( hydra, and(dragonet, exist( elemental, fire)))]))).
alc_query(and( hydra, and(dragonet, exist( elemental, fire)))).

15- ex2_1
====================
assert(tbox([equiv(a,or( b, c)), equiv(c,or( o, p)), equiv(b,or( m, n)), equiv(e,or( q, r)),
equiv(d,or( e ,f)), equiv(g,or( h ,i)), equiv(f,or( s ,t)), equiv(i,or( w ,x)), equiv(h,
or( u ,v1))])).
assert(sbox([ob(u@d,pred(p,[a])), ob(u@d,pred(p,[d])), ob(u@d,pred(p,[g])), ob(u@d,
pred(p,[~m])), ob(u@d,pred(p,[~n])), ob(u@d,pred(p,[~o])), ob(u@d,pred(p,[~p])), ob(u@d,
pred(p,[~q])), ob(u@d,pred(p,[~r])), ob(u@d,pred(p,[~s])), ob(u@d,pred(p,[~t])), ob(u@d,
pred(p,[~u])), ob(u@d,pred(p,[~v1])), ob(u@d,pred(p,[~w])), ob(u@d,pred(p,[~x]))])).
madl2_query(pe(u@d,pred(p,[and( a, and(d, and(g, and((~m), and((~n), and((~o), and((~p),
and((~q),and((~r), and((~s), and((~t),and( (~u), and((~v1), and((~w),(~x)))))))))))))))]))).
alc_query(and( a, and(d, and(g, and((~m), and((~n), and((~o), and((~p), and((~q),and((~r),
and((~s), and((~t),and( (~u), and((~v1), and((~w),(~x)))))))))))))))).
```

```
16- ex2_2
====================
assert(tbox([equiv(parentdog,or( papadog, mamadog)), equiv(husbanddog,and( maledog,
exist( haswife, femaledog))), equiv(maledog,and( dog,(~female))), equiv(papadog,and(
maledog, exist( haschild, dog))), equiv(femaledog,and( dog, female)), equiv(wifedog,
and( femaledog, exist(hashusband, maledog))), equiv(puppy,and( or( maledog, femaledog),
and(exist( hasmother, mamadog),and(exist( hasfather, papadog), (~exist(haschild, dog)
))))),equiv(mamadog,and( femaledog, exist( haschild, dog)))])).
assert(sbox([ob(u@d,pred(p,[wifedog])), ob(u@d, pred(p, [husbanddog]))])).
madl2_query(pe(u@d,pred(p,[and( wifedog, husbanddog)]))).
alc_query(and( wifedog, husbanddog)).

17- ex2_3
====================
assert(tbox([equiv(slitheringdragon,and( dragon, forall( transportmode, (~or( flying,
walking))))), equiv(walkingdragon,and( dragon, exist(transportmode, walking))),
equiv(firedrake,and( drake,and( forall( elemental, fire), exist(disposition, foe)))),
equiv(icedrake,and( drake,and(forall( elemental, water), exist(disposition, foe)))),
equiv(orientaldragon,and( walkingdragon, and(exist( elemental, water),forall(disposition,
friend)))), equiv(drake,and( walkingdragon, and(exist( elemental, or( water, fire)),
forall( disposition, foe)))), equiv(hydra,and( or( slitheringdragon, flyingdragon),
exist(disposition,foe))), equiv(westerndragon,and( flyingdragon,and( forall( elemental,
or(earth,water)), exist( disposition, foe)))), equiv(wyrm,and( slitheringdragon, exist(
elemental, water))), equiv(flyingdragon,and( dragon, exist(transportmode, flying))),
equiv(dragonet,and( forall( disposition, foe),and( or( walkingdragon,flyingdragon),
forall( elemental, (~or( earth,water))))))])).
assert(sbox([ob(u@d,pred(p,[dragonet])), ob(u@d,pred(p,[~and( forall( disposition, foe),
and(or( walkingdragon,flyingdragon), forall( elemental, (~or( earth, water)))))]))])).
madl2_query(pe(u@d,pred(p,[and( dragonet, (~and( forall( disposition, foe), and(or(
walkingdragon,flyingdragon), forall( elemental, (~or( earth, water)))))))]))).
alc_query(and( dragonet, (~and( forall( disposition, foe), and(or( walkingdragon,
flyingdragon), forall( elemental, (~or( earth, water)))))))).

18- ex2_4
====================
assert(tbox([equiv(b,and( (~a), a)), equiv(f,exist( p, and( g, h)))])).
assert(sbox([ob(u@d,pred(p,[exist( p, or( a, or(b, c)))])), ob(u@d, pred(p, [exist( q,
or( d, or(e, f)))])), ob(u@d,pred(p, [forall( p, and( c, d))]))])).
```

```
madl2_query(pe(u@d,pred(p,[and( exist( p, or( a, or(b, c))), and(exist( q, or( d,
or(e, f))),forall( p, and( c, d))))]))).
alc_query(and( exist( p, or( a, or(b, c))), and(exist( q, or( d, or(e, f))),forall( p,
and( c, d))))).

19- ex2_5
====================
assert(tbox([equiv(map,and( exist( contains, words), (~or( pictures, chapters)))),
equiv(nonfiction,and( (~fiction),and( exist( contains, chapters) ,forall(contains,
words)))), equiv(childrensbook,forall( contains, and( pictures, (~words)))), equiv(
fiction,or( map, and( chapters, words)))])).
assert(sbox([ob(u@d,pred(p,[nonfiction])), ob(u@d, pred(p,[childrensbook]))])).
madl2_query(pe(u@d,pred(p,[and( nonfiction, childrensbook)]))).
alc_query(and( nonfiction, childrensbook)).

20- ex2_7
====================
assert(sbox([ob(u@d,pred(p,[b])), ob(u@d, pred(p,[c])), ob(u@d, pred(p, [exist( p, and(
a, and(c, exist( r, (~d)))))])), ob(u@d, pred(p, [forall(r, d)]))])).
madl2_query(pe(u@d,pred(p,[and( b, and(c, and(exist( p, and( a, and(c, exist( r, (~d))))),
forall(r, d))))]))).
alc_query(and( b, and(c, and(exist( p, and( a, and(c, exist( r, (~d))))), forall(r, d))))).

21- ex2_8
====================
assert(tbox([equiv(b,( ~a))])).
assert(sbox([ob(u@d,pred(p,[forall( r, and( a, b))])), ob(u@d, pred(p, [exist( r, a)])),
ob(u@d, pred(p, [exist( r, b)]))])).
madl2_query(pe(u@d,pred(p,[and( forall( r, and( a, b)), and(exist( r, a), exist( r, b)))]))).
alc_query(and( forall( r, and( a, b)), and(exist( r, a), exist( r, b)))).

22- ex2_9
====================
assert(tbox([equiv(veggiepizza,and( pizza, forall( hastopping, (~meat)))), equiv(meatpizza,
and( pizza, forall( hastopping, (~veggie)))), equiv(veggie,or( mushroom, olive)),
equiv(meat,or( pepperoni ,sausage))])).
assert(sbox([ob(u@d,pred(p,[veggiepizza])), ob(u@d, pred(p, [meatpizza]))])).
madl2_query(pe(u@d,pred(p,[and( veggiepizza, meatpizza)]))).
alc_query(and( veggiepizza, meatpizza)).
```

```
23- ex3_1
====================
assert(tbox([equiv(dalek,and( biological, and(mechanical, forall( hasweakness,
blindness)))), equiv(monster,and( biological,and( (~humanoid), exist( hasweakness,
bullets)))), equiv(humanoid,and( biological,and( (~mechanical), forall( hasweakness,
bullets)))), equiv(cyberman,and( cyborg, forall( hasweakness, gold))), equiv(timelord,
and( biological, humanoid)), equiv(silurian,and( biological, forall( hasweakness,
bullets))), equiv(robot,and( mechanical, and( (~biological),and( forall( hasweakness,
logicalparadox), exist( hasweakness, sonicscrewdriver))))), equiv(quark,( robot)),
equiv(cyborg,and( mechanical,and( biological,and( exist( hasweakness,bullets),
exist( hasweakness, sonicscrewdriver)))))])).
assert(sbox([ob(u@d,pred(p,[dalek])), ob(u@d,pred(p,[cyborg])), ob(u@d,pred(p,
[cyberman])), ob(u@d,pred(p,[forall( hasweakness,sonicscrewdriver)])), ob(u@d,
pred(p,[silurian])), ob(u@d,pred(p,[monster])), ob(u@d,pred(p,[timelord])),
ob(u@d,pred(p,[forall(hasweakness, bullets)])), ob(u@d,pred(p,[quark])), ob(u@d,
pred(p,[exist( hasweakness, gold)])), ob(u@d,pred(p,[~and( and(timelord, monster),
and(and( timelord, dalek), and( quark, humanoid)))]))])).
madl2_query(pe(u@d,pred(p,[and( and(and( dalek, cyborg), and(and( cyberman, forall(
hasweakness,sonicscrewdriver)), and(and( silurian, monster), and(and( timelord,
forall(hasweakness, bullets)), and( quark, exist( hasweakness, gold)))))), (~and(
and(timelord, monster), and(and( timelord, dalek), and( quark, humanoid)))))]))).
alc_query(and( and(and( dalek, cyborg), and(and( cyberman, forall( hasweakness,
sonicscrewdriver)), and(and( silurian, monster), and(and( timelord, forall(hasweakness,
bullets)), and( quark, exist( hasweakness, gold)))))), (~and( and(timelord, monster),
and(and( timelord, dalek), and( quark, humanoid)))))).

24- ex3_2
====================
assert(tbox([equiv(slitheringdragon,and( dragon, forall( transportmode, (~or( flying,
walking))))), equiv(walkingdragon,and( dragon, exist( transportmode, walking))), equiv(
firedrake,and( drake,and( forall( elemental, fire), exist(disposition, foe)))), equiv(
icedrake,and( drake,and( forall( elemental, water), exist(disposition, foe)))),
equiv(orientaldragon,and( walkingdragon,and( exist( elemental, water),forall( disposition,
and(friend, exist( towards, or( people, animals))))))), equiv(drake,and( walkingdragon,
and(exist( elemental, or(water, fire)),forall(disposition, foe)))), equiv(hydra,and(or(
slitheringdragon, flyingdragon), exist( disposition,foe))), equiv(westerndragon,and(
flyingdragon,and( forall( elemental, or( earth,water)), exist( disposition, and( foe,
exist( towards, people)))))), equiv(wyrm,and( slitheringdragon, exist( elemental, water))),
equiv(flyingdragon,and( dragon, exist( transportmode, flying))), equiv(dragonet,and(
forall( disposition, foe),and( or( walkingdragon,flyingdragon), forall( elemental,
(~or( earth, water))))))])).
assert(sbox([ob(u@d,pred(p,[westerndragon])), ob(u@d, pred(p, [orientaldragon]))])).
madl2_query(pe(u@d,pred(p,[and( westerndragon, orientaldragon)]))).
alc_query(and( westerndragon, orientaldragon)).

25- ex3_3
====================
assert(tbox([equiv(parentdog,or(chepapadog,mamadog)), equiv(husbanddog,and(maledog,
exist( haswife, femaledog))), equiv(maledog,and( dog, (~female))), equiv(papadog,and(
maledog, exist( haschild, dog))), equiv(femaledog,and( dog, female)), equiv(wifedog,
and( femaledog, exist( hashusband, maledog))), equiv(puppy,and( or( maledog, femaledog),
and( exist( hasmother,mamadog),and( exist( hasfather, papadog), (~exist( haschild,
dog)))))), equiv(mamadog,and( femaledog, exist( haschild, dog)))])).
assert(sbox([ob(u@d,pred(p,[maledog])), ob(u@d,pred(p,[femaledog]))])).
madl2_query(pe(u@d,pred(p,[and( maledog, femaledog)]))).
alc_query(and( maledog, femaledog)).

26- ex3_4
====================
assert(tbox([equiv(a,( ~and( b, and(c, d)))), equiv(c,( ~or( f, (~f)))), equiv(b,( ~or(
e, (~e)))), equiv(d,( ~or( g, (~g))))])).
assert(sbox([ob(u@d,pred(p,[(~forall( p, a))])), ob(u@d, pred(p, [(~exist( p, (~a)))]))])).
madl2_query(pe(u@d,pred(p,[and( (~forall( p, a)), (~exist( p, (~a))))]))).
alc_query(and( (~forall( p, a)), (~exist( p, (~a))))).

27- ex3_5
====================
assert(tbox([equiv(g5,and( or( exist( uses, film), exist( has, zoom)), (~lcd))), equiv(
digslr,and( slr,and( digital, lcd))), equiv(slr,and( film ,zoom)), equiv(digital,and(
(~film) ,exist( has, zoom)))])).
assert(sbox([ob(u@d,pred(p,[g5])), ob(u@d, pred(p, [~digslr]))])).
madl2_query(pe(u@d,pred(p,[and( g5, (~digslr))]))).
alc_query(and( g5, (~digslr))).

28- ex3_7
====================
assert(sbox([ob(u@d,pred(p,[a])), ob(u@d, pred(p, [b])), ob(u@d, pred(p, [forall( p, c)])),
ob(u@d, pred(p, [forall( p, (~c))])), ob(u@d, pred(p, [exist( r,d)]))])).
madl2_query(pe(u@d,pred(p,[and( a, and(b, and(forall( p, c), and(forall( p, (~c)), exist
(r,d)))))]))).
alc_query(and( a, and(b, and(forall( p, c), and(forall( p, (~c)), exist( r,d)))))).

29- ex3_9
====================
assert(tbox([equiv(wierdopizza,and( pizza, exist( hastopping, alien))), equiv(meatpizza,
and( pizza, forall( hastopping, (~veggie)))), equiv(meat,or( pepperoni, sausage)),
equiv(veggie,or( mushroom, olive)), equiv(alien,( anchovy)), equiv(veggiepizza,and(
pizza, forall( hastopping, (~meat))))])).
assert(sbox([ob(u@d,pred(p,[~wierdopizza])), ob(u@d, pred(p,[veggiepizza]))])).
madl2_query(pe(u@d,pred(p,[or( (~wierdopizza), veggiepizza)]))).
alc_query(or( (~wierdopizza), veggiepizza)).

30- x_ex_aa
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)),
equiv(mother,and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,
person))), equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(
hasChild,father))), equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(
person,and(~brother,exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(
hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[brother]))])).
madl2_query(pe(u@d,pred(p,[sister]))).
alc_query(disjoint(brother,sister)).

31- x_ex_ab
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,and(
woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))), equiv(parent,
exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))), equiv(brother,
and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,exist(hasSibling,
person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[man]))])).
madl2_query(pe(u@d,pred(p,[woman]))).
alc_query(disjoint(man,woman)).

32- x_ex_ac
```

```
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)),
equiv(mother,and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,
person))), equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,
father))), equiv(brother,and(man,exist(hasSibling,person))),
equiv(sister,and(person,and(~brother,exist(hasSibling,person)))), equiv(luckyBrother,
and(man, forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[brother]))])).
madl2_query(pe(u@d,pred(p,[luckyBrother]))).
alc_query(subsum(luckyBrother,brother)).

33- x_ex_ad
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[parent]))])).
madl2_query(pe(u@d,pred(p,[father]))).
alc_query(subsum(father,parent)).

34- x_ex_ae
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)),
equiv(mother,and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,
person))), equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,
father))), equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,
and(~brother,exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,
sister)))])).
assert(sbox([ob(u@d,pred(p,[father]))])).
madl2_query(pe(u@d,pred(p,[grandfather]))).
alc_query(subsum(grandfather,father)).

35- x_ex_af
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[father]))])).
madl2_query(pe(u@d,pred(p,[grandfather]))).
alc_query(disjoint(grandfather,father)).

36- x_ex_ag
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)),equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild, person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,
father))), equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,
and(~brother,exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,
sister)))])).
assert(sbox([ob(u@d,pred(p,[sister]))])).
madl2_query(pe(u@d,pred(p,[grandfather]))).
alc_query(disjoint(grandfather,sister)).

37- x_ex_ah
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person, and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[a])), ob(u@d,pred(p,[b]))])).
madl2_query(pe(u@d,pred(p,[and(a,b)]))).
alc_query(and(a,b)).

38- x_ex_ai
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)),equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person, and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
madl2_query(pe(u@d,pred(p,[and(a,~a)]))).
alc_query(and(a,~a)).
```

```
39- x_ex_aj
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[a])), ob(u@d,pred(p,[b]))])).
madl2_query(pe(u@d,pred(p,[or(a,b)]))).
alc_query(or(a,b)).

40- x_ex_ak
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[a])), ob(u@d,pred(p,[b]))])).
madl2_query(pe(u@d,pred(p,[and(a,and(b,or(a,b)))]))).
alc_query(and(a,and(b,or(a,b)))).

41- x_ex_al
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[a])), ob(u@d,pred(p,[b]))])).
madl2_query(pe(u@d,pred(p,[and(a,and(b,or(a,~b)))]))).
alc_query(and(a,and(b,or(a,~b)))).

42- x_ex_am
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[a]))])).
madl2_query(pe(u@d,pred(p,[exist(r,a)]))).
alc_query(exist(r,a)).

43- x_ex_an
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)), equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist (hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[exist(r,a)])), ob(u@d,pred(p,[exist(r,~a)]))])).
madl2_query(pe(u@d,pred(p,[and(exist(r,a),exist(r,~a))]))).
alc_query(and(exist(r,a),exist(r,~a))).

44- x_ex_ao
====================
assert(tbox([equiv(man,and(person,male)), equiv(woman,and(person,~man)),equiv(mother,
and(woman,exist(hasChild,person))), equiv(father,and(man,exist(hasChild,person))),
equiv(parent,exist(hasChild,person)), equiv(grandfather,and(man,exist(hasChild,father))),
equiv(brother,and(man,exist(hasSibling,person))), equiv(sister,and(person,and(~brother,
exist(hasSibling,person)))), equiv(luckyBrother,and(man,forall(hasSibling,sister)))])).
assert(sbox([ob(u@d,pred(p,[forall(r,a)])), ob(u@d,pred(p,[exist(r,a)])), ob(u@d,
pred(p,[exist(r,~a)]))])).
madl2_query(pe(u@d,pred(p,[and(forall(r,a),and(exist(r,a),exist(r,~a)))]))).
alc_query(and(forall(r,a),and(exist(r,a),exist(r,~a)))).
```

Bibliography

[1]

[2]

[3] Martin Abadi. Logic in access control. In *Proceedings of the 18th Annual Symposium on Logic in Computer Science (LICS'03)*, pages 228–233, Ottawa, Canada, 2003. IEEE Computer Society Press.

[4] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A Calculus for Access Control in Distributed Systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.

[5] Sudhir Agarwal and Barbara Sprick. Specification of Access Control and Certification Policies for Semantic Web Services. In *Proceedings of the 6th International Conference on Electronic Commerce and Web Technologies (EC-Web 05)*, volume 3590 of *Lecture Notes in Computer Science (LNCS)*, pages 348–357, Copenhagen, 2005. Springer-Verlag.

[6] Sudhir Agarwal, Barbara Sprick, and Sandra Wortmann. Credential Based Access Control for Semantic Web Services. In *Proceedings of the 2004 American Association for Artificial Intelligence (AAAI) Spring Symposium*, volume 1, March 2004.

[7] Morteza Amini and Rasool Jalili. Multi-Level Authorization Model and Framework for Distributed Semantic-Aware Environments. *Accepted in IET Information Security*, 2010.

[8] P. E. Ammann and Ravi S. Sandhu. The Extended Schematic Protection Model. *Journal of Computer Security*, 1(3&4), 1992.

[9] Hajnal Andrka, Johan van Benthem, and Istvn Nmeti. Modal Languages and Bounded Fragments of Predicate Logic. Research Report ML-96-03, Institute of Mathematics, Hungarian Academy of Sciences, Budapest and Institute for Logic, Language and Computation, University of Amsterdam, 1996.

[10] ANSI. Information Technology - Role-Based Access Control. Technical report, American National Standards Institute, Inc., Febreuary 2004.

[11] L. Aqvist. Deontic Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 605–714. D. Reidel Publishing Company, 1984.

[12] Alberto Artosi, Paola Cattabriga, and Guido Governatori. KED: A Deontic Theorem Prover. In *Proceedings of the Workshop on Legal Application of Logic Programming (ICLP'94)*, pages 60–76, Firenze, Italy, 1994.

[13] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[14] Franz Baader and Philipp Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, page 452457, Sydney, Australia, 1991.

[15] Franz Baader, Ralf Kusters, and Frank Wolter. Extensions to Description Logic. In *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[16] Philippe Balbiani and Dimiter Vakarelov. PDL with intersection of programs: a complete axiomatization. *Journal of Applied Non-Classical Logics*, 13(3-4):231–276, 2003.

[17] Peter Balsiger and Alain Heuerding. Comparison of Theorem Provers for Modal Logics - Introduction and Summary. In *Proceedings of the Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Computer Science (LNCS)*, pages 25–26. Springer-Verlag, 1998.

[18] Steve Barker, Michael Leuschel, and Mauricio Varea. Efficient and Flexible Access Control via Logic Program Specialisation. In *Proceedings of the ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation (PEPM'04)*, pages 190–199, Verona, Italy, 2004. ACM Press.

[19] Steve Barker and Peter J. Stuckey. Flexible Access Control Policy Specification with Constraint Logic Programming. *ACM Transactions on Information and System Security (TISSEC)*, 6(4):501–546, 2003.

[20] D. E. Bell and L. J. Lapadula. Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, The Mitre Corporation, March 1976.

[21] Fikret Berkes. New and Not-So-New Directions in the Use of the Commons: Co-Management. *The Common Property Resoruce Digest*, 42:5–7, 1997.

[22] Elisa Bertino, Piero A. Bonatti, and Elena Ferrari. TRBAC: A Temporal Role-Based Access Control Model. *ACM Transaction on Information Systems Security*, 4(3):191–233, 2001.

[23] Elisa Bertino, Piero Andrea Bonatti, Elena Ferrari, and Maria Luisa Sapino. Temporal Authorization Bases: From Specification to Integration. *Journal of Computer Security*, 8(4):309–353, 2000.

[24] Elisa Bertino, Silvana Castano, and Elena Ferrari. Securing XML Documents with Author-X. *IEEE Internet Computing*, 5(3):21–31, 2001.

[25] Claudio Bettini, Sushil Jajodia, X. Sean Wang, and Duminda Wijesekera. Provisions and Obligations in Policy Management and Security Applications. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.

[26] Claudio Bettini, Sushil Jajodia, X. Sean Wang, and Duminda Wijesekera. Provisions and Obligations in Policy Rule Management. *Journal of Network and Systems Management*, 11(3):351–372, 2003.

[27] K. J. Biba. Integrity Considerations for Secure Computer Systems. Technical Report TR-3153, The Mitre Corporation, April 1977.

[28] Matt Bishop. *Computer Security: Art and Science*. Addison Wesley, 1 edition, Nov 2002.

[29] Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter. *Handbook of Modal Logic*. Elsevier, 2007.

[30] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2004.

[31] Piero A. Bonatti, Claudiu Duma, Norbert Fuchs, Wolfgang Nejdl, Daniel Olmedilla, Joachim Peer, and Nahid Shahmehri. Semantic Web Policies - A Discussion of Requirements and Research Issues. In *Proceedings of the 3rd European Semantic Web Conference (ESWC)*, volume 4011 of *Lecture Notes in Computer Science (LNCS)*, Budva, Montenegro, JUN 2006. Springer-Verlag.

[32] Piero A. Bonatti and Pierangela Samarati. *Logics for Emerging Applications of Databases*, chapter Logics for Authorizations and Security. Lecture Notes in Computer Science. Springer-Verlag, 2003.

[33] D. F. C. Brewer and M. J. Nash. The Chinese Wall Security Policy. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 215–228, Oakland, CA, USA, 1989.

[34] Diego Calvanese and Maurizio Lenzerini. Conjunctive Query Containment in Description Logics with n-ary Relations. In *Proceedings of the International Workshop on Description Logic (DL'97)*, volume 410 of *URA-CNRS*, pages 5–9, Paris, France, 1997.

[35] Silvana Castano, Maria Grazia Fugini, Giancarlo Martella, and Pireangela Samarati. *Database Security*. Addison-Wesley, 1996.

[36] L. Catach. Tableaux: A General Theorem Prover for Modal Logics. *Journal of Automated Reasoning*, 7(4):489–510, 1991.

[37] National Computer Security Center. Department of defense trusted computer system evaluation criteria (the orange book), 1985.

[38] Ming-Yen Chen, Ming-Fen Yang, Yuh-Ming Chen, and Hui-Chuan Chu. Development of a Semantic Awareness Framework for Textual Content Management in e-Learning. In *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, pages 429–430. IEEE Computer Society, 2006.

[39] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proceedings of the IEEE Computer Society Symposium on Security and Privacy*, pages 184–194, Oakland, CA, 1987.

[40] Oscar Corcho, Pinar Alper, Ioannis Kotsiopoulos, Paolo Missier, Sean Bechhofer, and Carole Goble. An overview of S-OGSA: A Reference Semantic Grid Architecture. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):102–115, 2006.

[41] Michael J. Covington, Matthew James Moyer, and Mustaque Ahamad. Generalized role-based access control for securing future applications. Technical report, College of Computing, Georgia Institute of Technology, 2000.

[42] Frdric Cuppens. An Epistemic and Deontic Logic for Reasoning about Computer Security. In *Proceedings of the European Symposium on Research in Computer Security*, pages 135–145, Toulouse, France, 1990.

[43] Frdric Cuppens. Roles and Deontic Logic. In *Proceedings of the Second International Workshop on Deontic Logic in Computer Science*, pages 86–106, Oslo, Norway, 1994.

[44] Frdric Cuppens and Robert Demolombe. A Deontic Logic for Reasoning about Confidentiality. In *Proceedings of the 3rd International Workshop on Deontic Logic in Computer Science*, pages 66–79, Sesimbra, Portugal, 1996.

[45] Mark Curphey, David Endler, William Hau, Steve Taylor, Tim Smith, Alex Russell, Gene McKenna, Richard Parke, Kevin McLaughlin, Nigel Tranter, Amit Klien, Dennis Groves, Izhar By-Gad, Sverre Huseby, Martin Eizner, Martin Eizner, and Roy McNamara. A Guide to Building Secure Web Applications: The Open Web Application Security Project. Technical report, 2002.

[46] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. A Fine-Grained Access Control System for XML Documents.

ACM Transactions on Information and System Security (TISSEC), 5(2):169–202, 2002.

[47] Nicodemos Damianou, Arosha K. Bandara, Morris Sloman, and Emil C Lupu. A survey of policy specification approaches. In *Handbook of Network and System Administration.* Elsevier, London, 2007.

[48] Mills Davis. Semantic Wave - Part 1. Technical Report A Project10X Special Report, Wilshire Conferences, Inc., 2006.

[49] Frank van Harmelen Deborah L. McGuinness. OWL Web Ontology Language Overview. Online, 2004. available at http://www.w3.org/TR/owl-features/.

[50] G. Denker, S. Nguyen, and A. Ton. OWL-S Semantics of Security Web Services: a Case Study. In *Proceedings of the 1st European Semantic Web Symposium*, pages 240–253, Heraklion, Greece, 2004.

[51] D. E. Denning. *Secure Information Flow in Computer Systems.* PhD thesis, Purdue Univeristy, 1975.

[52] D. E. Denning. Secure Distributed Data Views: The Sea-View Formal Security Model. Technical Report A003, SRI International, 1987.

[53] L.C. Dion. A Complete Protection Model. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 49–55, Oakland, CA, 1981.

[54] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating Datalog and Description Logics. *Intelligent Information Systems*, 10(3):227–252, 1998.

[55] Moussa A. Ehsan, Morteza Amini, and Rasool Jalili. A Semantic-based Access Control Mechanism Using Semantic Technologies. In *Proceedings of the 2nd International Conference on Security of Information and Networks (SIN 2009)*, Gazimagusa, North Cyprus, 2009.

[56] Sareh Sadat Emami, Morteza Amini, and Saadan Zokaei. A Context-Aware Access Control Model for Pervasive Computing Environments. In *Proceedings of the IEEE*

International Conference on Intelligent Pervasive Computing (IPC 2007), pages 51–56, Jeju Island, Korea, 2007.

[57] David Ferraiolo and Richard Kuhn. Role-Based Access Control. In *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, MD, 1992.

[58] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard for Role Based Access Control. *ACM Transactions on Information and System Securiy*, 4(3):224–274, 2001.

[59] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[60] Ian T. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *Proceeding of the IFIP International Conference on Network and Parallel Computing*, volume 3779 of *Lecture Notes in Computer Science (LNCS)*, pages 2–13. Springer-Verlag, 2005.

[61] Maria Grazia Fugini and Giancarlo Martella. ACTEN: A Conceptual Model for Security System Design. *Computers & Secuirty*, 3(3), 1984.

[62] Pedro Gama and Paulo Ferreira. Obligation Policies: An Enforcement Platform. In *Proceedings of the 6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)*, pages 203–212, Stockholm, Sweden, 2005.

[63] Olivier Gasquet and Bilal Said. Tableaux with Dynamic Filtration for Layered Modal Logics . In *Proceedings of the Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'07)*, volume 4548 of *Lecture Notes in Computer Science (LNCS)*, pages 107–118. Springer-Verlag, 2007.

[64] Rod Girle. *Modal Logics and Philosophy.* Acumen, 2000.

[65] Janice I. Glasgow, Glenn H. MacEwen, and Prakash Panangaden. Reasoning about Knowledge and Permission in Secure Distributed Systems. In *Proceedings of the First*

IEEE Computer Security Foundations Workshop (CSFW'88), pages 139–146, Franconia, New Hampshire, USA, 1988. MITRE Corporation Press.

[66] Randy Goebel, Sandra Zilles, Christoph Ringlstetter, Andreas Dengel, and Gunnar Grimnes. What Is the Role of the Semantic Layer Cake for Guiding the Use of Knowledge Representation and Machine Learning in the Development of the Semantic Web? In *Proceedings of the the 2008 AAAI Spring Symposium on Symbiotic Relationships between Semantic Web and Knowledge Engineering*, pages 45–50, Menlo Park, California, 2008. The AAAI Press.

[67] Robert Goldblatt. Mathematical Modal Logic: A View of its Evolution. *Journal of Applied Logic*, 1(5):309–392, 2003.

[68] Guido Governatori and Alessandro Luppi. Labelled Tableaux For Non-Normal Modal Logics. In *Proceedings of the 6th Congress of the Italian Association for Artificial Intelligence*, pages 119–130, Bologna, Italy, 2000.

[69] Erich Grädel. Decidable Fragments of First-Order and Fixed-Point Logic, From Prefix Vocabulary Classes to Guarded Logics. In *Proceedings of the Kalmr Workshop on Logic and Computer Science*, Szeged, Hungary, 2003.

[70] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the Decision Problem for Two-Variable First-Order Logic. *The Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[71] Erich Grädel and Martin Otto. On Logics with Two Variables. *Theoretical Computer Science*, 224(1-2):73–113, 1999.

[72] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proceedings of the 12th ACM international Conference on World Wide Web (WWW'03)*, pages 48–57, Budapest, Hungary, 2003. ACM.

[73] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[74] Rajeev Gupta, Shourya Roy, and Manish Bhide. Identity Delegation in Policy Based Systems. In *Proceedings of the 8th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY '07)*, pages 229–240, Bologna, Italy, 2007. IEEE Computer Society.

[75] V. Haarslev and R. Moller. RACER System Description. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR'01)*, volume 2083 of *Lecture Notes in Computer Science (LNCS)*, pages 701–705, Siena, Italy, 2001. Springer-Verlag.

[76] Reiner Hähnle. Tableaux And Related Methods. In John Alan Robinson and Andrei Voronkov, editors, *Handbook Of Automated Reasoning*, pages 100–178. Elsevier and MIT Press, 2001.

[77] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in Operating Systems. *Communications of ACM*, 19(8):461–471, 1976.

[78] Urs Hengartner and Peter Steenkiste. Exploiting Information Relationships for Access Control. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM'05)*, pages 269–278, Washington, DC, USA, 2005. IEEE Computer Society.

[79] Leon Henkin. The Completeness of the First-Order Functional Calculus. *Symbolic Logic*, 14(3):159–166, 1949.

[80] Thomas Herchenroder. *Lightweight Semantic Web Oriented Reasoning in Prolog: Tableaux Inference for Description Logics*. PhD thesis, University of Edinburgh, 2006.

[81] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1996.

[82] ISO/IEC:. *Information Technology - Open Systems Interconnection - Security Frameworks for Open Systems: Access Control Framework. ISO/IEC 10181-3*, Nov 1995.

[83] ISO/IEC:9594-8. ITU-T Recommendation X.509: Infomation Technology - Open Systems Interconnection - The Directory : Public-Key and Attribute Certificate Frameworks. Technical report, ITU-T, 2001.

[84] Jafar Haadi Jafarian and Morteza Amini. CAMAC: A Context-Aware Mandatory Access Control Model. *ISeCure- The ISC International Journal of Information Security*, 1(1):35–54, 2009.

[85] Jafar Haadi Jafarian, Morteza Amini, and Rasool Jalili. A Context-Aware Mandatory Access Control Model for Multilevel Security Environments. In *Proceedings of the 27th International Conference on Computer Safety, Reliability and Security (SafeComp 2008)*, volume 5219 of *Lecture Notes in Computer Science (LNCS)*, pages 401–414, Newcastle, UK, 2008. Springer.

[86] S. Jajodia and Ravi S. Sandhu. Toward a Multilevel Relational Data Model. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 50–59, Denver, Colorado, 1991.

[87] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible Support for Multiple Access Control Policies. *ACM Transactions on Database Systems*, 26(2):214–260, 2001.

[88] Sushil Jajodia, Pierangela Samarati, and V. S. Subrahmanian. A Logical Language for Expressing Authorizations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 31–42, Oakland, CA, USA, 1997.

[89] Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and Elisa Bertino. A Unified Framework for Enforcing Multiple Access Control Policies. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 474–485, Tucson, AZ, USA, 1997. ACM Press.

[90] Sara Javanmardi, Morteza Amini, and Rasool Jalili. An Access Control Model for Protecting Semantic Web Resources. In *Proceedings of the 2nd International Semantic Web Policy Workshop (SWPW'06) 2006*, pages 32–46, Athens, GA, USA, 2006.

[91] Sara Javanmardi, Morteza Amini, Rasool Jalili, and Yaser GanjiSaffar. SBAC: A Semantic Based Access Control Model. In *Proceedings of the 11th Nordic Workshop on Secure IT-systems (NordSec2006)*, pages 157–168, Linkping, Sweden, 2006.

[92] Robert Johnson. *Parallel Analytic Tableaux Systems.* PhD thesis, Queen Mary and Westfield College, University of London, 1996.

[93] Audun Josang, Dieter Gollmann, and Richard Au. A Method for Access Authorisation Through Delegation Networks. In *Proceedings of the Australasian Workshops on Grid Computing and E-Research (ACSW Frontiers '06)*, pages 165–174, Hobart, Tasmania, Australia, 2006. Australian Computer Society, Inc.

[94] James B. D. Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A Generalized Temporal Role-Based Access Control Model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, 2005.

[95] Audun Jsang, Roslan Ismailb, and Colin Boydb. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2):618–644, 2007.

[96] Lalana Kagal, Tim Finin, and A. Joshi. A Policy-Based Approach to Security for the Semantic Web. In *Proceedings of the 2nd International Semantic Web Conference (ISWC03)*, Sanibel Island, Florida, USA, Oct 2003.

[97] Alan H. Karp. Authorization-Based Access Control for the Services Oriented Architecture. In *Proceedings of the 4th International Conference on Creating, Connecting, and Collaborating through Computing (C5)*, Berkeley, CA, USA, 2006.

[98] Saket Kaushik, Duminda Wijesekera, and Paul Ammann. Policy-Based Dissemination of Partial Web-Ontologies. In *Proceedings of the 2005 Workshop on Secure Web Services (SWS '05)*, pages 43–52, Fairfax, VA, USA, 2005. ACM Press.

[99] M. Kudo. PBAC: Provision-based Access Control Model. *International Journal of Information Security*, 1(2):116–130, Feb 2002.

[100] Butler Lampson. Protection. In *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, pages 437–443, Princeton University, 1971.

[101] Alon Y. Levy and Marie-Christine Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.

[102] Juan Li. *Semantics-Based Resource Discovery in Global-Scale Grids*. PhD thesis, The University of British Columbia, 2008.

[103] Ninghui Li. *Delegation Logic: A Logic-based Approach to Distributed Authorization*. PhD thesis, New York University, 2000.

[104] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation Logic: A Logic-based Approach to Distributed Authorization. *ACM Transactions on Information Systems Security*, 6(1):128–171, 2003.

[105] Zhen Li. An Implementation of a Tableau Theorem Prover for Modal Logics. In *Proceedings of the Second International Joint Conference on Automated Reasoning (IJCAR'04)*, Cork, County Cork, Ireland, 2004.

[106] Thorsten Liebig and Felix Müller. Parallelizing Tableaux-Based Description Logic Reasoning. In *Proceedings of the International Workshops on the Move to Meaningful Internet Systems (OTM'07)*, volume 4806 of *Lecture Notes in Computer Science (LNCS)*, pages 1135–1144, Vilamoura, Portugal, 2007. Springer-Verlag.

[107] R. J. Lipton and L. Snyder. A Time Linear Algorithm for Deciding Security. *Journal of the ACM*, 24(3):455 – 464, 1977.

[108] Zhen Liu, Anand Ranganathan, and Anton Riabov. Specifying and Enforcing High-Level Semantic Obligation Policies. In *Proceedings of the 8th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*, pages 119–128, Bologna, Italy, 2007.

[109] Zhen Liu, Anand Ranganathan, and Anton Riabov. Specifying and Enforcing High-Level Semantic Obligation Policies. In *Proceedings of the 8th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*, pages 119–128, Bologna, Italy, 2007.

[110] Emil C. Lupu and Morris Sloman. Conflicts in Policy-Based Distributed Systems Management. *IEEE Transactions on Software Engineering*, 25(6):852–869, Nov/Dec 1999.

[111] Carsten Lutz. NEXPTIME-Complete Description Logics With Concrete Domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[112] Carsten Lutz, Holger Sturm, Frank Wolter, and Michael Zakharyaschev. A Tableau Decision Algorithm for Modalized ALC with Constant Domains. *Studia Logica*, 72(2):199 – 232, 2002.

[113] AmirReza Masoumzadeh, Morteza Amini, and Rasool Jalili. Context-Aware Provisional Access Control. In *Proceedings of the 2nd International Conference on Information Systems Security (ICISS'06)*, volume 4332 of *Lecture Notes in Computer Science (LNCS)*, pages 132–146, Kolkata, India, 2006. Springer-Verlag.

[114] AmirReza Masoumzadeh, Morteza Amini, and Rasool Jalili. Conflict Detection and Resolution in a Context-Aware Authorization System. In *Proceedings of the 3rd IEEE Symposium on Security in Networks and Distributed Systems (SSNDS'07)*, pages 505–511, Niagara Falls, Canada, 2007.

[115] D. Harrison McKnight and Norman L. Chervany. The Meanings of Trust. Technical report misrc, University of Minnesota, Management Information Systems Reseach Center, 1996.

[116] Paul McNamara. Deontic Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2006.

[117] Jing Mei, Zuoquan Lin, and Harold Boley. $\mathcal{ALC}^u_{\mathbb{P}}$: An Integration of Description Logic and General Rules. In *Proceedings of the First International Conference on Web Reasoning and Rule Systems*, volume 4524 of *Lecture Notes in Computer Science (LNCS)*, pages 163–177, Innsbruck , Austria, 2007.

[118] Marco Casassa Mont and Robert Thyne. A Systemic Approach to Automate Privacy Policy Enforcement in Enterprises. In *Proceedings of the 6th Workshop on Privacy Enhancing Technologies*, volume 4258 of *Lecture Notes in Computer Science (LNCS)*, pages 118–134, Cambridge, UK, 2006. Springer.

[119] M. Mortimer. On Language with Two Variables. *Zeit. für Math. Logik und Grund. der Math*, 21:135–140, 1975.

[120] Tim Moses. eXtensible Access Control Markup Language, Version 2.0. Technical report, OASIS Standard, 2005.

[121] Tim Moses. *eXtensible Access Control Markup Language, Version 2.0*. OASIS Standard, Feb 2005.

[122] Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):4160, 2005.

[123] Michael J. Murphy, Michael Dick, and Thomas Fischer. Towards the Semantic Grid: A State of the Art Survey of Semantic Web Services and their Applicability to Collaborative Design, Engineering, and Procurement. *Journal of Communications of the IIMA*, 8(3):11–24, 2008.

[124] Anton Naumenko. *Semantics-Based Access Control in Business Networks*. PhD thesis, University of Jyvasky, 2007.

[125] Linh Anh Nguyen. Analytic Tableau Systems for Propositional Bimodal Logics of Knowledge and Belief. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'02)*, volume 2381 of *Lecture Notes In Computer Science (LNCS)*, pages 206–220, Copenhagen, Denmark, 2002. Springer-Verlag.

[126] Matunda Nyanchama and Sylvia L. Osborn. The Role Graph Model and Conflict of Interest. *ACM Transaction on Information Systems Security*, 2(1):3–33, 1999.

[127] Vineet Padmanabhan and Guido Governatori. On Constructing Fibred Tableaux for BDI Logics. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence (PRICAI'06)*, pages 150–160, Guilin, Guanxi, China, 2006.

[128] J. Park and R.S. Sandhu. The uconabc usage control model. *ACM Transactions on Information and System Security*, 7(1):128–174, 2004.

[129] Laura Pearlman, Von Welch, Ian Foster, Carl Kesselman, and Steven Tuecke. A Community Authorization Service for Group Collaboration. In *Proceedings of the 3rd IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'02)*, pages 50–59, Monterey, CA, USA, 2002. IEEE Computer Society.

[130] Torsten Priebe, Wolfgang Dobmeier, Christian Schlager, and Nora Kamprath. Supporting Attribute-based Access Control in Authorization and Authentication Infrastructures with Ontologies. *Journal of Software*, 2(1):27–38, 2007.

[131] Eric Prud'hommeaux. W3C ACL System. Technical report, The World Wide Web Consortium (W3C), 2004.

[132] Li Qin and Vijayalakshmi Atluri. Concept-Level Access Control for The Semantic Web. In *Proceedings of the 2003 ACM Workshop on XML Security (XMLSEC'03)*, pages 94–103, New York, NY, USA, 2003. ACM Press.

[133] Fausto Rabitti, Elisa Bertino, Won Kim, and Darrell Woelk. A Model of Authorization for Next-Generation Database Systems. *ACM Transactions on Database Systems (TODS)*, 16(1):88–131, 1991.

[134] Pavan Reddivari, Tim Finin, and Anupam Joshi. Policy-Based Access Control for an RDF Store. In *Proceedings of the IJCAI-07 Workshop on Semantic Web for Collaborative Knowledge Acquisition*, 2007.

[135] D. De Roure. Future for European Grids: GRIDs and Service Oriented Knowledge Utilities. 2009. `http://www.semanticgrid.org/documents/ngg3/ngg3.html`.

[136] Véronique Royer and J. Joachim Quantz. Deriving Inference Rules for Terminological Logics. In *Proceedings of the European Workshop on Logics in Artificial Intelligence (JELIA'92)*, volume 633 of *Lecture Notes in Computer Science (LNCS)*, pages 84–105, Berlin, Germany, 1992. Springer Verlag.

[137] John M. Rushby, Sam Owre, and Natarajan Shankar. Subtypes for Specifications: Predicate Subtyping in PVS. *IEEE Transactions on Software Engineering*, 24(9):709–720, 1998.

[138] Pierangela Samarati and Sushil Jajodia. Data Security. Technical report, Supported by DARPA/Rome Laboratory under contract F30602-96-C-0337, 1998.

[139] Ravi S. Sandhu. A Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Scheme. *Journal of ACM*, 35(2), 1988.

[140] Ravi S. Sandhu. The Typed Access Matrix Model. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy (SP'92)*, pages 122–136, Oakland, CA , USA, 1992.

[141] Ravi S. Sandhu, Venkata Bhamidipati, Edward J. Coyne, Srinivas Ganta, and Charles E. Youman. The ARBAC97 Model for Role-Based Administration of Roles: Preliminary Description and Outline. In *Proceedings of the ACM Workshop on Role-Based Access Control*, pages 41–50, 1997.

[142] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[143] Ravi S. Sandhu and Pierrangela Samarati. Access Control: Principles and Practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.

[144] D. Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:477, 1962.

[145] Haibo Shen and Fan Hong. A Context-Aware Role-Based Access Control Model for Web Services. In *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE 2005)*, pages 220– 223, 2005.

[146] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A Practical OWL-DL Reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[147] Ken P. Smith and Marianne S. Winslett. Entity Modeling in the MLS Relational Model. In *Proceedings of the 18th Conference on Very Large Databases*, pages 199–210, Vancouver, Canada, 1992. Morgan-Kaufmann.

[148] Masakazu Soshi. Safety Analysis of the Dynamic-Typed Access Matrix Model. In *Proceedings of the 6th European Symposium on Research in Computer Security*, volume 1895 of *Lecture Notes In Computer Science (LNCS)*, pages 106–121. Springer-Verlag, 2000.

[149] William Stallings. *Cryptography and Network Security Principles and Practice.* Prentice Hall, 2003.

[150] Gerald Stermsek, Mark Strembeck, and Gustaf Neumann. Using Subject- and Object-Specific Attributes for Access Control in Web-based Knowledge Management Systems. In *Proceedings of the Workshop on Secure Knowledge Management (SKM'04)*, Amherst, NY, USA, 2004.

[151] Virginie Thion, Serenella Cerrito, and Marta Cialdea Mayer. A General Theorem Prover for Quantified Modal Logics. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'02)*, volume 2381 of *Lecture Notes In Computer Science (LNCS)*, pages 266 – 280, Copenhagen, Denmark, 2002. Springer-Verlag.

[152] Alessandra Toninelli, Rebecca Montanari, Lalana Kagal, and Ora Lassila. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In *Proceedings of the 5th International Semantic Web Conference (ISWC'06)*, volume 4273 of *Lecture Notes in Computer Science (LNCS)*, pages 473–486, Athens, GA, USA, 2006. Springer-Verlag.

[153] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Computer Science (LNCS)*, pages 292–297, Seattle, Washington, USA, 2006. Springer-Verlag.

[154] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, page 93, Washington, DC, USA, 2003. IEEE Computer Society.

[155] A. Uszok, J. M. Bradshaw, J. Lott, M. Breedy, L. Bunch, P. Feltovich, M. Johnson, and H. Jung. New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS. In *Proceedings of the IEEE Workshop on Policies for Distributed Systems and Networks (Policy'08)*, pages 145–152, Palisades, NY, USA, 2008. IEEE Press.

[156] Andrzej Uszok, Jeffrey M. Bradshaw, Matthew Johnson, Renia Jeffers, Austin Tate, Jeff Dalton, and Stuart Aitken. KAoS Policy Management for Semantic Web Services. *IEEE Intelligent Systems*, 19(4):32–41, 2004.

[157] Daniel J. Weitzner, Jim Hendler, Tim Berners-Lee, and Dan Connolly. Creating a Policy-Aware Web: Discretionary, Rule-based Access for the World Wide Web. In lena Ferrari and Bhavani Thuraisingham, editors, *Web and Information Security.* IOS Press, Hershey, PA, USA, 2004.

[158] Duminda Wijesekera and Sushil Jajodia. A Propositional Policy Algebra for Access Control. *ACM Transactions on Information and System Security*, 6(2):286–325, 2003.

[159] Thomas Y. C. Woo and Simon S. Lam. Authorizations in Distributed Systems: A New Approach. *Journal of Computer Security*, 2(2-3):107–136, 1993.

[160] Zhaohui Wu and Huajun Chen. *Semantic Grid: Model, Methodology, and Applications.* Advanced Topics in Science and Technology in China. Springer-Verlag, 2008.

[161] Mariemma Yague, Maria del Mar Gallardo, and Antonio Mana. Semantic Access Control Model: A Formal Specification. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *Lecture Notes on Computer Science (LNCS)*, pages 24–43, Milan, Italy, 2005. Springer-Verlag.

[162] Mariemma I. Yague, Antonio Mana, Javier Lopez, and Jose M. Troya. Applying the Semantic Web Layers to Access Control. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA '03)*, pages 622–626, Prague, Czech Republic, 2003. IEEE Computer Society.

[163] Anis Yousefi, Rasool Jalili, and Mahdi Niamanesh. Multi-Determiner Protection of Private Data in Pervasive Computing Environments. *IJCSNS International Journal of Computer Science and Netwrok Security*, 6(12):239–248, 2006.

[164] Guangsen Zhang and Manish Parashar. Context-Aware Dynamic Access Control for Pervasive Applications. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, USA, 2004.

[165] Xiao Ming Zhang. A Semantic Grid Oriented to E-Tourism. In *Proceedings of the First International Conference on Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science (LNCS)*, pages 485–496, Beijing, China, 2009. Springer-Verlag.

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Rasool Jalili) Principal Adviser

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Ali Movaghar)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Hassan Mirian)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Mohammad Ardeshir)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Saeed Jalili)

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy.

(Dr Mehran Soleiman Fallah)

Approved for the University Committee on Graduate Studies