Formal methods applied to access control policy design, verification and enforcement

Clara Bertolissi

HABILITATION À DIRIGER DES RECHERCHES
DE
L'UNIVERSITÉ AIX-MARSEILLE

MÉMOIRE

en vue de l'obtention d'une

HABILITATION À DIRIGER DES RECHERCHES

Spécialité : **Informatique**

présentée par

Clara BERTOLISSI

Formal methods applied to access control policy design, verification and enforcement

---

Soutenue le **25 11 2022** devant le jury composé de :

M.	Alessandro	**ARMANDO**	Universitá di Genova	Rapporteur
Mme	Olga	**KOUCHNARENKO**	Institut FEMTO-ST	Rapporteuse
Mme	Maribel	**FERNANDEZ**	King's College London	Examinatrice
Mme	Alessia	**MILANI**	Université Aix-Marseille	Examinatrice
M.	Benjamin	**NGUYEN**	INSA Centre Val de Loire	Rapporteur
M.	Jean-Marc	**TALBOT**	Université Aix-Marseille	Examinateur

Résumé

L'évolution des systèmes d'information classiques a introduit des nouvelles technologies et des nouveaux services pour la gestion et le partage d'informations. Avec la diffusion d'Internet, il est désormais possible de partager facilement de grandes quantités d'informations électroniques et de ressources informatiques dans des environnements distribués ouverts. Ces environnements servent de plateforme commune pour des utilisateurs hétérogènes (par exemple, les entreprises, les particuliers, etc.) en hébergeant des services, des applications utilisateur et des données. Cette évolution a introduit des nouvelles menaces pour la sécurité de l'information et des nouveaux défis pour la modélisation du contrôle d'accès. Afin de relever ces défis, de nombreux travaux de recherche étudient l'extension des modèles traditionnels de contrôle d'accès vers des modèles conçus pour gérer l'accès aux données et la définition des privilèges dans les systèmes distribués et dynamiques.

Dans ce manuscrit, nous contribuons à cette ligne de recherche en introduisant un formalisme pour la spécification de politiques de contrôle d'accès interopérables et développées de manière autonome dans différents domaines administratifs. La formalisation à base de règles que nous proposons permet l'analyse et la validation des politiques par la vérification des propriétés telles que la consistance et l'efficacité, ainsi que la simulation de scénarios d'exécution. Nous avons également étudié le contrôle d'accès dans le contexte des systèmes avec une prise de décision collaborative. Nous proposons notamment une solution pour le contrôle d'accès dans les services web cooperatifs avec des dépendances transitives, ainsi que dans les processus de fusion de données.

Enfin, nous considérons les politiques d'autorisation et les contraintes liées aux systèmes de gestion de flux de travail (*workflows*) et leurs applications. Nous proposons une méthodologie pour surveiller un workflow et sa politique de contrôle d'accès associée, afin d'en assurer la terminaison en respectant toutes les contraintes de sécurité ou, si ce n'est pas possible, une exécution avec une violation minimale de ces contraintes.

Abstract

The evolution of classical information systems has introduced new technologies and services for information managing and sharing. With the spread of the Internet, it is now possible to easily share vast amounts of electronic information and computer resources in open distributed environments. These environments serve as a common platform for heterogeneous users by hosting services, user applications and data. This evolution has opened new threats to information security and new challenges to access control modeling. In order to meet these challenges, many research works went towards extending traditional access control models towards models tailored for managing data access and privilege definition within distributed and dynamic systems.

In this manuscript, we contribute to this line of research by introducing a modular framework for the specification of interoperable access control policies developed autonomously in different administrative domains. The rule-based formalisation we propose allows for policy analysis and validation by verification of policy properties and simulation of execution scenarios. We have also investigated access control in the context of collaborative decision making systems. In particular, we propose a solution for controlling access in cooperating web services with transitive dependencies, as well as in data fusion processes. Finally, we consider authorization policies and constraints related to workflow management systems and workflow-driven applications. We propose a methodology to monitor a workflow and its associated access control policy in order to ensure a successful termination of the workflow or, if not possible, an execution with a minimal violation of the authorization constraints.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The challenges of access control and management in traditional information systems have been resolved through the proposal of several access control models such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC). But in nowadays society, technological evolution brings new challenges for governing access within dynamic contexts. The growing complexity of our socio-economic environment produces large systems, most often dynamic, distributed, large scale and sometimes composed of other systems. The complexity of interoperable, inter-connected systems lies in the managerial and operational independence of their components, in their geographical distribution, and in their evolutionary development.

In [GQ96] two principles were put forward for integrating access control specifications of separate administrative domains: *autonomy* - i.e. if an access is permitted within an individual system, it must also be permitted in the federated system; and *security* - if an access is not permitted within an individual system, it must not be permitted in the federated system. Given the scale and complexity of today distributed systems, satisfying both autonomy and security is a daunting task requiring to develop

(i) a modular framework for the abstract specification of interoperable access control policies developed autonomously in different administrative domains,

(ii) an automated and efficient analysis technique for the validation of access control policies in various execution scenarios, and

(iii) a sound methodology to enforce access control policies based on standard modules that can be easily re-used or integrated into existing solutions.

Requirement (i) refers to the expressiveness of the framework which should permit the specification of authorization conditions based on attributes of subjects and resources - beyond identity and role - and on context information such as time and location. Additionally, it is crucial to provide specification constructs capable of associating access control policies with each cooperating system as well as supporting their combination with mechanisms guaranteeing the satisfaction of the principles of autonomy and security.

Testing distributed systems is well-known to be computationally expensive [Gar97]. This observation justifies requirement (ii) above which, when combined with (i), poses the challenge of finding the best trade off between expressiveness – i.e. the framework should permit the specification of the largest set of policies of practical interest – and amenability to analysis – i.e. the framework should support efficient techniques for the validation of policies so to help policy designers to gain confidence in their correctness already at design time. Depending on the context, formal verification uses a wide variety of models and techniques, such as automata, rewriting systems, type systems, abstract interpretation, constraints, process calculi, to cite a few. Following the formal methods approach, systems are analyzed from a semantic point of view (system behavior, properties to ensure, model of the attacker) and guarantees to be provided are identified.

Moreover, with formal techniques, there is no need to implement a prototype to experiment with the system behavior, which may be quite difficult in the case of composed systems. The simulation of an abstract specification of the system does not require the availability of the environment in which the system will be deployed, it is enough to model those aspects which are relevant to taking security access decisions and then invoke the available analysis procedure.

Indeed, an approach satisfying requirements (i) and (ii) is only a preliminary step in the direc-

tion of building secure systems. In fact, as stipulated by requirement (iii), run-time techniques for enforcing access policies are essential for the secure deployment of systems and have been receiving considerable attention from the research community; see, e.g., [BBG06]. Ideally, it should be possible to refine the abstract and validated specification of the access control policies into policies that can be readily enforced by an access control mechanism developed on top of available and well-engineered modules which can be easily combined together.

In this manuscript, I present some contributions that address the three requirements identified above in different contexts, namely open distributed environments, collaborating systems and business processes. The manuscript is organised as follows:

Chapter 2 presents the preliminary notions on access control policies and models, as well as on term rewriting and its application to security policy specification and analysis.

Chapter 3 introduces a rewrite-based unifying framework based on the notion of *category* called CBAC. Several access control models can be understood in category-based terms, e.g. variants of RBAC, including hierarchical, time and location aspects, as well as lattice-based models. We propose a distributed version of CBAC for modelling (and enforcing) global access control policies that take into account the local policies specified and maintained by each component of a distributed system. In particular we ensure the coherence of a global access control decision w.r.t. local access control requirements by specifying in a tunable way how to integrate access authorisations resulting from the local policies. The declarative approach we adopt permits properties of access control policies to be proved in a modular way. We show how consistency and totality properties of access control policies can be derived from standard properties of the rewrite framework we use and how narrowing techniques can be used to help administrators debugging their policies.

Chapter 4 considers access control in the context of cooperative systems. In this setting, data coming from multiple autonomous sources is processed into new pieces of information that can be further processed by other entities participating in the cooperation. Controlling

the access to such evolving and variegated data, often under the authority of different entities, is challenging. We present two solutions in the context of web services and data fusion processes.

Web services are a form of distributed cooperating system architecture and seem to become the preferred implementation technology for realising the integration and interaction between various systems in the Internet. In this context, a particularly difficult case occurs when a service invokes another service to satisfy an initial request, leading to indirect authorization errors. To overcome this problem, we propose a new approach based on a version of ORganization Based Access Control (OrBAC) extended by a delegation graph to keep track of transitive authorization dependencies. We give an axiomatisation of our model and, following the original OrBAC specification, we choose Datalog as specification language. As a byproduct of choosing Datalog, our framework supports the automated analysis of execution scenarios through the invocation of available Datalog engines. We also show how to implement an enforcement mechanism for our model by extending the standard XACML architecture.

In multi-source cooperative systems, methods for collaborative decision making while ensuring an appropriate level of control to the different parties involved are needed. In our work, we identify a set of access control requirements and propose an attribute-based access control model where provenance information is used to specify access constraints that account for both the evolution of data objects and the process of data fusion. We also demonstrate the feasibility of the proposed model by showing how it can be implemented in existing XACML-based access control frameworks by adding an external module for dealing with Provenance information.

Chapter 5 is dedicated to the security properties of business processes and in particular to workflow management systems and workflow-driven applications, which need to mediate access to their resources by enforcing authorization policies and constraints, such as Separation of Duty. We propose a new methodology to build run-time monitors capable of ensuring the successful termination of workflows subject to authorization constraints, i.e. capable of answering user requests to execute tasks in a workflow, in such a way that the policy is not violated and the workflow instance is guaranteed to terminate. The methodology is based

on state-of-the-art Satisfiability Modulo Theories techniques and composed of an off-line pre-computation phase to generate a reachability graph representing all the possible terminating executions, and an on-line phase, refining the set of solutions keeping into account the actual security constraints. An extensive experimental evaluation with an implementation of the technique shows the scalability of the proposed approach. Among the existing related problems, such as finding execution scenarios that not only satisfy the workflow but also satisfy other properties (e.g., that a workflow instance is still satisfiable even in the absence of users), we have solved the problem of finding the set of solutions minimizing a suitably defined notion of cost/risk of violating a constraint, when no completely secure executions exists (i.e. executions satisfying all security constraints).

Chapter6 concludes with a summary of the presented work and the related perspectives. A discussion on future research directions is also developed.

This manuscript presents some research I have done since the defense of my PhD thesis during my visiting periods in King's College London and in FBK Trento and as Maître de Conférences in Aix-Marseille University, with several co-authors. I refer to the corresponding published articles in each chapter. Several recent works are not included in this manuscript, namely the work presenting an administration model for access control policies [BFT20], the definition of a graph-based language providing a graphical representation of policies [BFT21], and a framework for modular composition of (administrative) access control policies [BF22].

Chapter 2

Preliminary notions

2.1 Access control : definition and historical background

The access control objective is to control computational resources and digital information to prevent unauthorized disclosure (confidentiality) and improper malicious modifications (integrity), while ensuring access for authorized entities (availability).

A first step in the development of an access control model is the identification of the objects o to be protected, the subjects s that execute activities and request access to objects, and the actions a that can be executed on the objects, and that must be controlled. An authorization can then be represented as a triple *(s, o, a)*, indicating that authorization subject s can execute action a over authorization object o.

The implementation of access control is based on three main concepts corresponding to a conceptual separation between different levels of abstraction [DFJS07, SdV01]

Access control policies define the rules according to which access control must be regulated. In general, access control policies are dynamic in nature as they have to reflect evolving business factors, government regulations, and environmental conditions. Policies are high-level requirements that specify how access is managed and who may access information under what circumstances. The definition of access control policies (and their corresponding models) is far

from being a trivial process. One of the major difficulties lies in the interpretation and application of different real world security rules (often complex and sometimes ambiguous) coming, for example, from practices and organizational regulations. A security policy must capture all the different rules to be enforced and translate them in well defined and unambiguous rules enforceable by a computer system.

Access control models provide formal representation of access control security policies. The formalization allows the proof of the security properties that are provided by the designed access control system. We will give more details more specifically on rule-based access control models in Section 2.3.

Access control mechanisms usually come at the low abstraction level where they enforce these high-level access control policies and translate a user's access request in terms of a specific structure that the system provides. Access control models are essential for bridging the abstraction gap between access policies and mechanisms. In particular, the separation between policies and mechanisms introduces an independence between protection requirements to be enforced and mechanisms enforcing them. This helps discussing protection requirements independently of their implementation and allows one to compare different access control policies as well as different mechanisms that enforce the same policy. Indeed, if a mechanism is tied to a specific policy, a change in the policy would require changing the whole access control system; mechanisms able to enforce multiple policies avoid this drawback. The implementation of a correct mechanism is far from being trivial: mapping access control primitives to a computer system is a complex task and moreover possible security weaknesses may be due to the implementation itself. The access control mechanism must work as a reference monitor, that is, a trusted component intercepting each and every request to the system.

The terminology "access control" has its origins in the transportation literature of the first half of the twentieth century [Pre47]. The first form of limited-access roads, also referred to as "controlled access" highways, such as the Bronx River Parkway, dates back to 1907. By forcing cars to enter and exit via one-way ramps, controlled access highways reduce the probability of cross-traffic accidents. The 60s saw access control primarily as a mechanism to ensure availabil-

ity of the system. As the multi-user computer systems progressed, confidentiality and integrity quickly became important, particularly in computer systems within the military and intelligence services. We see the introduction at that time of the new notion of "authorization": the system restricted access to users who were authorized, meaning that individuals could be sanctioned by the organization in case of illegal access. The literature of the 1970s reflects an increasing interest in computer security. Government agencies wanted assurances that sensitive data would be protected from compromise during computation. Early papers of this period use "protection" and "access control" interchangeably. Lampson's work of 1971, "Protection" [Lam71], defines protection as: "a general term for all the mechanisms that control the access of a program to other things in the system", including for instance a supervisor/user mode and access control by user to file directories. Among the first and most famous access-control models from this period, there is Mandatory Access Control, or MAC. A multi-level security version of MAC was described in the Department of Defense Trusted Computer System Evaluation Criteria [oD85]. In MAC systems there is a central policy administrator who defines whether a user is allowed to access a particular object. The user must first identify her/hisself to the system via authentication. The access request is mediated by a reference monitor, which queries the authorization policy defined by the administrator to determine whether the access should be allowed. An alternative model to MAC, not assuming that the access-control policy is managed by a central authority, is Discretionary Access Control (or DAC). As Jordan summarized in 1987 [Jor87], DAC is "discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject". After the introduction of MAC and DAC, the predominant access control Model, introduced in the literature by Sandhu in 1996, is Role-Based Access Control (RBAC) [SCFY96, FSG+01]. This approach aims to reduce the overhead necessary for managing permissions for large numbers of users by first assigning users to one or more roles, and then assigning permissions to those roles (rather than individual users). At the beginning of the 2000s, the National Institute of Standards and Technology (NIST) commissioned a study of RBAC and its economic impact [OL10], which found that it is "arguably the most important innovation in identity and access management since discretionary and mandatory access control". From there, the literature describing extensions to

the basic RBAC model is extensive. Some examples include efforts to model the introduction of obligations [PS04], and the delegation of privileges [WKB07, ZOS03] as well as the introduction of the notion of temporal [JBLG05] or geographic [BCDP05] constraints in RBAC policies. Several other models, such as task-based access control [TS97], organisation-based access control [KBB+03], risk-based access control [McG09, dSMS+16], event-based access control [BFB07] or provenance-based access control [PNS12] have been proposed in the last twenty years, resulting in a variety of models tailored to the specific needs of particular systems and their applications. More recently, some research efforts have also been done in defining a comprehensive unifying framework where different models can be accommodated (see more about his in Chapter 3).

2.2 Towards new models of access control

Traditional access control solutions do not respond adequately to new challenges addressed by modern computer systems. Today highly-distributed, network-connected, heterogeneous and open computing environments require fine-grained and flexible models for protecting the access and usage of digital resources. We describe in this section three settings that we have considered in our research work.

2.2.1 Open distributed environments

With the development of the Internet, the fundamentals of access control need to be rethought. Whereas traditional access control systems were based on the identity of the party requesting a resource, in open environments this approach cannot be applied because often the requestor and the resource belong to different domains which do not know each other. Therefore, a more appropriate approach consist of using properties (attributes) of the requestor and of the resource as a basis for access decision. In the last years, Attribute-based access control [JKS12b, YT05, HKF15] has been developed and differs from the traditional discretionary access control model by replacing the subject by a set of attributes and objects by properties associated with them.

The use of attributes in access-control policies in the Internet environment is exemplified by XML-based access control languages, such as WS-Policy [D.03] and the extensible Access Control Markup Language (XACML) [OAS13]. An important advantage of XML-based access control languages is the interoperability, that consists in the possibility of exchanging policies through different systems or services using the same access control language. As reported in [DDS05], this feature is particularly interesting in an open environment like the Internet, where a single system, which has to be protected as a single entity, may be distributed over the Net. Initially, XML-based access control languages were thought only for the protection of resources that were themselves XML files [DDCdVPS02, GB02, Gab04]. Recent proposals instead use XML to define languages for expressing protection requirements on any kind of data/resources [spe03, ADS04]. XACML is the most common XML-based access control language used. It allows policy-writers to specify arbitrary attributes of users and resources and write rich rules limiting access based on different combinations of attributes. XACML 1.0 was ratified as an OASIS standard in 2003 and the latest version 3.0 was ratified in January 2013. The XACML standard covers three major parts - the reference architecture, the policy language and the request/response protocol.

The increasing need of new web applications easily accessible over the Internet makes the security issues even more challenging in such a distributed and heterogeneous world. The most common technique for developing web applications is the use of web services and web service composition. In [YT02] the authors propose a semantic approach for access control in web services and define an XML-based access control language called Semantic Policy Language (SPL) for the description of access control criteria based on the use of attribute certificates. SPL is modular, enables the abstraction and reuse of components and the composition of SPL policies in an unambiguous way. In [BSPM06] the authors define a fine grained access control systems for web services called Ws-AC. Their model comes with a service with (formally defined) negotiation capabilities and their policies can be translated into the WS-Policy standard. Focusing on web service composition, in [Yin07] authors propose an access control model defined in a declarative policy specification language which uses pure-past linear temporal logic (PPL TL). In their approach, past histories of service invocations can be used to make access

control decisions. Moreover, they support parametric access control rules, role hierarchies and separation of duty constraints.

A different approach which has received some interest in the scientific community is the Usage Control (UCON) model proposed by R. Sandhu and J. Park in 2002 [PS02] that marks a shift from access to usage control. UCON main features are mutability of attributes and continuity of access decisions evaluation, which are especially important for dynamic and open environments like the Web. The main novelty w.r.t. previous models is in the way access decision are computed. The access decision taken over particular conditions should be recalculated as soon as initial conditions change to avoid security breaches in the system. This means that if during the ongoing access attributes change and the security policy is not satisfied any more, the UCON authorization system revokes granted rights and terminates the resource usage. The security policy is enforced via a usage control reference monitor [PHB+08] consisting of three basic components: an authorizations module, a conditions module and an obligations module. The access decision is a conjunction of decisions evaluated by every module. Usage control has been implemented in various environments, e.g operating systems [KB07], GRID computing and service oriented architectures [BAB+07, PMH07].

2.2.2 Collaborating component systems

Nowadays, enterprises are frequently faced with the demand to integrate different (possibly heterogeneous) systems which are independently designed and developed. One of the most challenging issues in federated systems is to find satisfactory answers to the question of how component systems can efficiently cooperate while preserving their autonomous characteristics. When considering security, autonomy includes authentication and authorisation [BBG06] for which several proposals have been put forward under the name of Federated Identity Management (FIdM) [Cha09]. Identity Management refers to the policies, processes, and technologies that establish user identities and enforce rules about access to digital resources. With an enterprise/organization identity management system, rather than having separate credentials for each system, a user can employ a single digital identity to acces all resources to which the user

is entitled. FIdM permits extending this approach above the enterprise level, creating a trusted authority for digital identities across multiple organizations. Many FIdM initiatives have been developed by major industrial players such as Microsoft, IBM, Sun Microsystems, and Oracle. In the literature several solutions [BD04, Bha06, BBG06, CB04, GHHF05, MKT05, McA04, SAS04, WW06, BBG07, LK07, Pha10] have been proposed for both federated authentication and federated authorization management. However, industrial and academic FIdM solutions have different levels of maturity when considering authentication or authorization. Authentication has been widely studied and robust implementations based on the Single Sign-On (SSO) protocol [Hug05, ACCP12] are routinely used in many real-world systems for supporting authentication across multiple domains while eliminating the need to maintain distinct user credentials for separate applications. Authorisation instead is still a problem which is not fully addressed. Mostly of the authorization models for federated systems use appropriately developed authorisation policy languages, such as Ponder [DDLS01] or X-FEDERATE [BBG06]. One of the most important challenges is how to compare and evaluate the differences of similar or dissimilar policies from different domains. The difference in understanding organisational factors such as roles and positions contributes to the complexity of the challenges. This difference also leads to certain difficulties in managing constraints in the delegation.

A first issue to be addressed is where the authorization policy should be defined and by whom it should be administered. In general, following [SdV01], the authorisation policy can be controlled via three approaches: i) by the federation authority [JD94] This approach helps reduce the administrative cost but challenges the mandatory security autonomy of the member domain—, ii) by the member domain authority [Cas97] This approach could lead to inconsistency in the authorisation policy between the member domains —, iii) or by both (hybrid approach) [DCdVS96]. There should be a balance between the power of the federation and the local authority so that the burden of the authorisation process is not distributed too heavily on either side.

Moreover, if we take the point of view of data being exchanged and used in the process of collaboration, another issue is the definition of the access control policies related to these data, which may be the result of fusing information coming from different entities. A few works

propose a solution to this problem by using policy templates [dHZ16a] or provenance-based policies [BdHZ19].

2.2.3 Business Processes

Business processes capture the activities that must be performed in a business setting to provide a service or product. They are modeled by workflows, defined in the Cambridge dictionary as "the way that a particular type of work is organised, or the order of the stages in a particular work process". Workflows thus represent structured collections of tasks to be executed in the view of completing the desired goal. In contrast with access control policies discussed previously, here we do not consider atomic isolated actions, but rather actions as part of a more complex process, where they are organized following an execution (partial) order. Security-related dependencies are specified as authorization policies — which user is entitled to do which task — and additional authorization constraints (such as Separation of Duties) on the execution of the various tasks. The enforcement of authorization policies and constraints is fundamental for security [LRM14], however it may lead to deadlock situations where a workflow instance cannot be completed because no task can be executed without violating either the authorization policy or the constraints. Researchers have been addressing this problem under the name of workflow satisfiability problem (WSP) [WL10]. The WSP is inherently Hard. Several variations of the WSP and similar problems have been defined in the literature and there are many solution methods available. A survey of the work done on these problems may be found in [dS17]. Indeed, the relations between workflows and authorizations are not as obvious as they may seem at first. A workflow specification spans at least three perspectives: control-flow, data-flow, and authorization [vdAtHKB03]. Control-flow constrains the execution order of the tasks (e.g., sequential, parallel, or alternative execution); the data-flow defines the various data objects consumed or produced by these tasks; and the authorization specifies the organizational actors responsible for the execution of the tasks in the form of authorization policies and constraints. These three dimensions are interconnected, as each one of them influences the others. The set of behaviors (i.e., possible executions of the workflow) allowed by the control-flow is fur-

ther constrained by conditions on the data, as well as by user assignments and constraints in the authorization perspective. There are several categories of control-flow support: sequential execution of tasks (e.g. [BFA99]; parallel executions (e.g., [CGGJ16]); and others like CSP [Hoa85] and Petri nets [Mur89], which add support for conditional branches and loops (e.g., [BBK11]). There are also several classes of authorization constraints for workflows that have been identified in the literature, such as counting constraints, entailement constraints and user-independent constraints. For instance, the SoD constraints are user-independent.

Given the conflicting goals of business compliance (the business processes must follow the workflow model and satisfy all the associated constraints) and business continuity (business must not stop, even in adverse conditions, e.g. no available users) defining good trade-offs has been a topic of research in the Business Process management and security communities. A common practice in the analysis of workflow satisfiability is to abstract away from parts of a workflow specification. For instance, few works take into account the data-flow (some completely disregard it, e.g. [CGGJ16], and some model it with non-deterministic decisions, e.g., [BBK11]). It is also usual practice to limit the allowed control-flow constructs and supported authorization constraints.

Some works [CGKW17, BdSR18], address the problem of *Weighted* WSP where solutions for minimizing weights, to be interpreted as costs of violating a security constraint, are proposed.

2.3 Rule-based policy specification and analysis

Rule-based policy specifications have the advantage to be concise and easy to maintain for security administrators. Several rule-based frameworks for access control have been proposed in the recent years [JSSS01, HW08, KBB+03, BS04], to cite a few. These works propose formal, rule-based languages for policies. Usually this formal basis is a logic suitable to express the dynamic access control conditions for a given application domain. PROLOG and Datalog are often used to obtain answer sets from a database of distributed policies. The negative aspect is that when using declarative approaches it could be extremely difficult to have a formal

"meaning" for every set of access control policies, such that one can compute this meaning and inspect whether it is the same as the policy author's intention.

First-order logic has proven to be sufficient for representing historically important access control models (e.g., an access control matrix can be viewed as a conjunction of propositions). However, there are access control policy requirements for which classical logic is not sufficient for specification. Modal logics have been employed by Abadi [Aba03, ABLP93] to reason about additional features of access control. Other works use deontic logic [CC97], defeasible logic [LBOG06], description logics [ZHLL05], fibred logic [BBGG09], etc.

Other rule-based approaches use term rewriting. Echahed and Prost in [EP05] used rewrite rules for the definition of the functions that control the confidentiality level of data, more specifically by describing how security levels are downgraded. The work concerns concurrent programs, whose formal model relies on a variant of process calculus. In [BF06], authors model access control lists and role based access control as term rewrite systems. They give a rewriting interpretation to the properties of a RBAC policy in the same direction as presented in [DKKdO07, San08], where in addition a reduction strategy is used for request evaluation.

In [JM06, Mor07], authors propose a formalism to describe security policies as functions on state transitions. Systems must ensure that a given security predicate defined by the policy is true along all their execution. In such framework, it is possible to distinguish the policy statement from its implementation, which is particularly useful for comparing different implementations of the same policy. A similar approach is taken in [TK06] where authors compare different approaches following their formal model of policies, including XACML and the logic language of [HW08].

A natural question is to compare the expressive power of (first-order) rewriting and (first-order) logic. Since both paradigms are Turing complete, from the theoretical point of view, decidability results are a concern in both approaches. In the logic-based setting, authors point out some characterizations of decidable subsets of first order logic, for example in [HW08]. For term rewriting, there exists several classes of rewrite systems for which important properties are decidable (see below and e.g. [Ter03]).

We believe rewriting is an interesting approach for defining policies in a declarative manner, and can suitably be viewed as an intermediate language for policies. An advantage of term rewriting systems is the number of available efficient implementations, which allows fast prototyping, and the tools and methods that allows one to check properties like termination, for example. Following such approach, we are able to characterize properties of policies with respect to the properties of the rewriting systems defining them. Unfortunately, all these properties are undecidable in general.

A first important property concerns whether a policy will ever return a result for a given access request. This is an important question, since policy rules often involve some computation or deduction prior to taking a decision and such process should be finite.

Termination *Termination* is surely an issue for rewriting based policies when compared to classic Datalog (without negation or constraints), for which all programs terminate. However, rewriting provides a lot of flexibility in policy specification, where terms can have a "deep" structure, in contrast with Datalog. Quite powerful techniques to check termination for term rewriting systems are available, such as recursive path orderings [Der87], semantic labeling [Zan95], dependency pairs [AG00], etc. These and other techniques have been implemented by several tools that can (statically) verify termination for a large set of rewrite systems: for example AProVe [GSKT06], TTT [HM05], and CiME [MU04], to cite a few.

Next, we introduce totality, which means that for every access request, there exists a corresponding access decision (e.g. authorization or prohibition).

Totality With the term *Totality*, we mean the property ensuring that for every access request, there exists a corresponding access decision. Totality in rewrite-based policies corresponds to the notion of sufficient completeness of a rewrite system. It states that every ground term evaluates to a term exclusively built with constructors (and possibly variables) [Com86, KNRZ91]. Several algorithms have been developed to check sufficient completeness, or to complete a set of patterns so that this property is satisfied [Bou94]. The soundness and completeness of a

policy can be checked using tools like CiME [19], by analysing the normal forms of access requests. Sufficient completeness is decidable for terminating and confluent term rewriting systems [BJ12], and the complexity of the decision procedures varies with the kind of rewrite system in question, ranging from NP-complete for free constructor systems to exp-time on linear systems [KNRZ91].

Other typical queries relate to the property of policy consistency.

Consistency A security policy is *consistent* if it computes at most one access decision for a given input request. This properties corresponds to the confluence of the underlying rewrite system. For terminating systems, it is sufficient to check confluence locally, by testing the joinability of critical pairs [KB83]. Interestingly, in this case the process for checking the consistency of policies can be mechanized via the completion algorithm [KB83]. A left linear rewrite system which does not have any critical pair is orthogonal. Orthogonal rewrite systems are not necessarily terminating, but they are always confluent [Hue80].

Besides proving more fundamental properties like consistency, termination and totality, other policy properties may be of particular interest to policy administrators, such as

- safety: whether a given user can acquire a particular permission in a given situation;
- liveness: whether all resources are protected by some authorisation rule;
- effectiveness: whether there exists principals not assigned to any permission, or there exists resources which are not accessible.

As for the previous properties, their verification is not always decidable. These properties are related to the notion of reachability. For verifying a property, one tests for the reachability of system states representing unwanted situations. If such kind of states cannot be reached by the set of derivations of a given system, then the security flaw will never occur. A number of analysis techniques have been developed for user-role reachability analysis of ARBAC [SYRG07, JLT+08, AR12]. In [DFK06], the authors suggest to perform goal reachability over Datalog

programs as a way to compare access control policies. Reachability over term rewriting is a very general approach which constructs approximations of the set of normal forms with respect to a rewrite system, see for instance [Gen98].

Given the potentially critical consequences of policy updates, it is also important for administrators to analyse the effects of a change in the policy, A (sequence of) administrative action(s) may cause profound transformations in the behaviour of the policy. Change-impact analysis has previously been studied for instance for firewall policies in [Liu07] and for RBAC policies using a model checking approach in [FKMT05]. For policies specified as term rewriting systems, narrowing techniques can be used to perform static analysis of administrative update actions. In [BTV16], narrowing is used to compute counter-examples to the equivalence of rewrite-based policies. Narrowing can also be used for executing review queries: an administrator can review e.g. the capabilities associated with principals, by choosing an appropriate query term and looking at its narrowing derivation tree.

Access control policy analysis may rely also on a graph-based approach. In [JT01], access control is defined via binary relations on graphs, creating a model which is basically a derivation of RBAC. The complexity of the specifications is controlled with the help of some additional constraint types, such that the safety problem can be verified. Whereas in [KMPP04], safety is shown decidable for an access control model based on graphs, with some restrictions on the form of the graph transformation rules. The decidability results rely on existing theorems for the decidability properties for graph transformations. Authors show the usability of their approach on RBAC. If a policy is represented as a graph, and policy updates are modelled using graph rewriting, one could compare the graph before and after the rewriting step. The formalism proposed in [AF17, BFT21] combines the use of a visual graph formalism to represent a concrete state of the system, and the use of rewrite rules to model the dynamics of the system. In [KMPP01], the authors propose a framework based on the theory of graph transformations for the description of the evolution of a policy and the comparison of different policy models (i.e. DAC and Lattice-based models).

Chapter 3

A unified access control meta-model for distributed environments

With the increasing interest in access control over the years, a variety of models and languages have been proposed. In accessing a given resource, a policy may dictate that a user has a need-to-know, is appropriately cleared, is competent, has not already performed a different operation on the same resource, is incapable of accessing other enterprise resources, or is capable of accessing an object while performing a specific task. In this rich context, consisting of a wide range of different policy models, Atluri and Ferraiolo in [FA08] raised the question on the prospects for and benefits of a meta-model of access control. As an example of past tentatives, the authors mention the work in [FGHK05], where NIST had proposed an access control framework, referred to as the Policy Machine (PM) that has been shown to accommodate a wide variety of access control policies including DAC, MAC, and RBAC. In later work [FAG11], the PM has been refined for specifying and enforcing a wide variety of attribute-based policies. At another level, the XACML policy specification language, focusing on the interoperability among applications, has been growing in recognition and use.

In response to [FA08], the following year Barker in [Bar09], argues that existing access control models are based on a small number of primitive notions that can often simply be specialized for domain-specific applications. A similar consideration was made by Landin [Lan66] in the

context of programming languages: rather than developing n special programming languages for n application areas, it is essential instead to identify a set of programming "primitives" from which a specific subset may serve as a basis for deriving a particular language (for a particular area of application). Similarly, multiple existing access control models can be expressed in terms of a set of primitive notions, and many "novel" access control models can potentially be developed by simply combining the primitives of access control in novel ways.

3.1 Synthesis of our approach

Continuing on this line of research, in [BF10b] we focus on the primitives of access control models and we propose a formal framework for their representation. The first primitive notion that we identify as common to access control models is the categorization of principals. Informally, a category is a group to which entities may be assigned. Classification types used in access control, like classifications by role, user attributes, status, clearances, discrete measures of trust, team membership, location, etc, can be seen as particular instances of the more general notion of category. Thus, categorization in our meta-model can be seen as an extension of the concept of user groups, a feature usually supported even by early approaches of access control. Like groups, categories of users can be nested and need not be disjoint. Support of categories greatly simplifies management of authorizations, since a single authorization granted to a category can be enjoyed by all its members. Categories can be organised into hierarchical relationships supporting authorization implication and thus clearly simplifying authorization management.

The other primitive features we consider are methods for describing properties and relationships between principals, and for specifying modalities like permissions and authorizations. We formalize in [BF10a] all of the notions that underlie our approach by developing a rewrite-based framework later extended to distibuted environments [BF11]. The use of a rewrite specification allows us to ensure a clean and unambiguous semantics; moreover, the possibility to write policies as modular sets of authorisation rules offers administrators more flexibility and

simplicity for specifying and combining access control policies [BF08b].

To demonstrate the expressive power of the category-based meta-model, we show how a range of access control models can be defined as specific instances of the meta-model. In particular, we show how variants of RBAC, including hierarchical, time and location aspects, as well as lattice-based models such as the Bell-Lapadula and the McLean models can be specified. We also describe how a number of novel access control models can be derived as particular cases of the meta-model.

Traditional access control frameworks generally assume a single monolithic specification of the entire access control policy. However, in many real world situations, access control needs to combine requirements independently stated that should be enforced as one, while retaining their administrative autonomy. Starting from these observations, it is clear the need of a policy composition framework by which different component policies can be integrated while retaining their independence. In [BF14] we extend the meta-model to incorporate the notion of component policy. We use site identifiers to denote contextual or environmental information e.g., IP addresses, times, system states, components identifiers, external states, etc. The notion of distributed environment that we consider is related to the notion of federation developed in the context of database systems (see for example [JD94, dVS97], where a federated system integrates several databases while preserving their autonomy). We propose a formal framework for modelling (and enforcing) global access control policies that take into account the local policies specified and maintained by each member of the federation. Our idea of categorisation applies also to this kind of distributed, federative settings. In a system with dispersed resources, classification of entities may depend on the site to which the entity belongs. Moreover, permissions associated to categories of entities may also depend on the site where the category is defined. When multiple policy modules (e.g., for different authorities or different domains) exist for the specification of access control rules, the access control system should provide a means for users to specify how the different modules should interact, for example, if their union (maximum privilege) or their intersection (minimum privilege) should be considered. Using the techniques introduced in [BF09], we ensure the coherence of a global access control decision w.r.t. local access control requirements by specifying in a tunable way how to integrate access

authorisations resulting from the local policies.

In addition to permissions, we define a notion of forbidden operation (or banned action) on resources. When both permissions and denials can be specified, the problem naturally arises of how to deal with incompleteness (accesses for which no rule is specified) and inconsistency (accesses for which both a denial and a permission are specified). A relation "undeterminate" can be defined if authorisations and prohibitions are not complete, i.e., if there are access requests that are neither authorised nor denied (thus producing an undeterminate answer in a three-valued policy). These notions are essential for integrating partially specified policies, i.e. policies that may be "not applicable" to requests on resources that are out of their jurisdiction, either in a centralised or distributed access control system. Dealing with inconsistencies requires support for conflict resolution policies.

In the distributed metamodel we can combine evaluations from different sources in a flexible way by refining the definition of the function evaluating the global authorisation. Many conflict resolution techniques can be applied. Standard operators to compose policies, such as Intersection, Subtraction, Union, Precedence operator can be easily defined in the metamodel. Moreover, the combination algorithms of the XACML policy language [OAS03, LWQ+09], such as Permit-overrides, Deny-overrides, First-applicable, Only-one-applicable can be specified in our framework using recursive rules (for a complete specification see [BF08b]). To facilitate the declarative definition of more sophisticated operators, it is useful to add higher-order features to the specification language. In [BF08b, BF14] we describe a higher-order extension of the language in order to include policy combination operators such as override, closure and scope restriction (as defined e.g. in [BdCdVS00, WJ03a]) and further policy combination expressions including such operators. Indeed, different approaches can be taken to deal with conflict resolution. If it is true that some solutions may appear more natural than others, none of them represents "the perfect solution" applicable in all situations.

The declarative approach we adopt permits properties of access control policies to be proved in a modular way. In particular, as already presented in Section2.3, we are interested in consistency and totality properties of policies. These properties guarantee that access requests will be

treated as expected. We show how consistency and totality properties of access control policies can be derived from standard properties of the rewrite framework we use. We define a notion of "safe" system as sufficient conditions for these properties to hold. The general rules of the meta-model satisfy these (syntactic) conditions by construction. This means that to ensure that a policy specification obtained as an instance of the metamodel is consistent and total it is sufficient to check that the rules defining the functions specific to that instance satisfy the safeness conditions. This can be done manually or (semi)-automatically by using rewriting tools such as CiME [CPU+10] or AproVe [GSKT06]. In [BU13a] a prototype is developed allowing policy designers to specify and test their access control policy. Since the policy implementation follows the metamodel general schema, the prototype provides an intuitive means of specifying a wide range of policies. We have developed a banking use case following both a centralised and a distributed scenario. After having specified the policy, the security administrator can choose via the interface which security property he wants to test. The translation of the specification into the rule-based syntax is executed in a transparent way and the tool CiME3 is launched in order to perform automated analysis. The results obtained on the rewrite systems are then translated into natural language using security policy terms to ease the readability for the policy designer. For some properties (such as termination), if the tests are successful, CiME3 is able to provide a formal certification by producing a trace of the proof in the form of a Coq certificate.

Contributions: summarising, the main contributions of this work are the following.

- a declarative, rewrite-based *specification* of a distributed, category-based access control metamodel, where distributed systems are seen as federations in which each component preserves its autonomy;
- a technique to define *combinations* of policies, by defining general policy-combining operators;
- a formal operational semantics for *access request evaluation* in centralised as well as in distributed contexts where information is shared, including mechanisms for the resolution

of conflicts between local and global policies;

- a technique to prove *totality and consistency* of access control policies, based on termination and confluence properties of the underlying term rewriting system.

This chapter is based on the results published in [BF10a, BF11, BF08a, BF08b, BF09, BF14, BTV16, BU13b].

3.2 The CBAC meta-model

We briefly describe below the key concepts underlying the category-based metamodel of access control, henceforth denoted by $\mathcal{M}$. We refer the reader to [Bar09] for a detailed description.

The CBAC metamodel consists of a family $\mathcal{E}$ of sets of **entities**, which are classified into **categories**, a family $\mathcal{R}el$ of **relationships between entities**, and a set $\mathcal{A}x$ of **axioms** that specify the properties that the model must satisfy.

Informally, a category is any of several distinct classes or groups to which entities may be assigned. Entities are denoted uniquely by constants in a many sorted domain of discourse, including:

- A countable set $\mathcal{C}$ of *categories*, denoted c_0, c_1, ...
- A countable set $\mathcal{P}$ of *principals*, denoted p_0, p_1, ...
- A countable set $\mathcal{A}$ of named *actions*, denoted a_0, a_1, ...
- A countable set $\mathcal{R}$ of *resource identifiers*, denoted r_0, r_1, ...
- A finite set $\mathcal{A}uth$ of possible *answers* to access requests.

  Additionally, the following sets can be introduced in specific models:

  - A countable set $\mathcal{E}$ of *event identifiers*, denoted e_0, e_1, ...
  - A countable set $\mathcal{S}$ of *situational identifiers*.

In addition to the different types of entities mentioned above, the metamodel includes the following relations that are of primary importance for the specification of access control policies:

- *Principal-category assignment:* $\mathcal{PCA} \subseteq \mathcal{P} \times \mathcal{C}$, such that $(p, c) \in \mathcal{PCA}$ iff a principal $p \in \mathcal{P}$ is assigned to the category $c \in \mathcal{C}$.
- *Permissions:* $\mathcal{ARCA} \subseteq \mathcal{A} \times \mathcal{R} \times \mathcal{C}$, such that $(a, r, c) \in \mathcal{ARCA}$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ can be performed by principals assigned to the category $c \in \mathcal{C}$.
- *Authorisations:* $\mathcal{PAR} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$, such that $(p, a, r) \in \mathcal{PAR}$ iff a principal $p \in \mathcal{P}$ can perform the action $a \in \mathcal{A}$ on the resource $r \in \mathcal{R}$.

Thus, $\mathcal{PAR}$ defines the set of authorisations that hold according to an access control policy that specifies $\mathcal{PCA}$ and $\mathcal{ARCA}$.

Definition 3.1 (Axioms) *The relation $\mathcal{PAR}$ satisfies the following core axiom, where we assume that there exists a reflexive-transitive relation $\subseteq$ between categories; this can simply be equality, set inclusion (the set of principals assigned to $c \in \mathcal{C}$ is a subset of the set of principals assigned to $c' \in \mathcal{C}$), or an application specific relation defining a hierarchy of categories may be used. If $c \subseteq c'$ we say that c is above c', and c' is below c; for example* manager $\subseteq$ employee.

$$(a1)\quad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},$$
$$(\exists c \in \mathcal{C}, \exists c' \in \mathcal{C}, (p, c) \in \mathcal{PCA} \wedge c \subseteq c' \wedge (a, r, c') \in \mathcal{ARCA}) \Leftrightarrow (p, a, r) \in \mathcal{PAR}$$

The category-based metamodel of access control CBAC is based on the core axiom $(a1)$ for $\mathcal{PAR}$ given in Def. 3.1.

Definition 3.2 (Policy) *Given a specification of $\mathcal{E}$ and $\mathcal{R}el$, and a set $\mathcal{A}x$ of axioms defining an access control model M, a policy is a tuple $\langle E, Rel \rangle$ that defines the contents of $\mathcal{E}$ and $\mathcal{R}el$ such that E, Rel satisfy all the axioms in $\mathcal{A}x$.*

Operationally, this axiom can be realised through a set of function definitions [BF10a], as summarised below.

Definition 3.3 *The information contained in the relations $\mathcal{PCA}$ and $\mathcal{ARCA}$ is modelled by the functions* pca *and* arca*, respectively, where* pca *returns the list of categories assigned to a principal, e.g.* $\mathsf{pca}(p) \rightarrow [\mathsf{c}]$*, and* arca *returns a list of permissions assigned to a category, e.g.* $\mathsf{arca}(c) \rightarrow [(\mathsf{a}_1, \mathsf{r}_1), \ldots, (\mathsf{a}_n, \mathsf{r}_n)]$.

The rewrite-based specification of the axiom $(a1)$ *in Def. 3.1 is given by the rewrite rule:*

$$(a2) \quad \mathsf{par}(p, a, r) \quad \rightarrow \quad if \ (a, r) \in \mathsf{arca}^*(\mathsf{below}(\mathsf{pca}(p))) \ then \ \mathsf{grant} \ else \ \mathsf{deny}$$

Briefly, pca returns the list of categories assigned to a principal, e.g. $\mathsf{pca}(\mathsf{p}) \rightarrow [\mathsf{c}]$, and arca returns a list of permissions assigned to a category, e.g. $\mathsf{arca}(\mathsf{c}) \rightarrow [(\mathsf{a}_1, \mathsf{r}_1), \ldots, (\mathsf{a}_n, \mathsf{r}_n)]$. As the function name suggests, below computes the set of categories that are below (*w.r.t.* the hierarchy defined by the $\subseteq$ relation) any of the categories given in the list $\mathsf{pca}(P)$. For example, for a given category c, this could be achieved by using a rewrite rule $\mathsf{below}([\mathsf{c}]) \rightarrow [\mathsf{c}, \mathsf{c}_1, \ldots, \mathsf{c}_n]$. The function $\in$ is a membership operator on lists, grant and deny are answers, and arca^* generalises the function arca to take into account lists of categories.

An access request by a principal p to perform the action a on the resource r can then be evaluated simply by rewriting the term $\mathsf{par}(\mathsf{p}, \mathsf{a}, \mathsf{r})$ to normal form.

Note that $(a1)$, and its algebraic version $(a2)$, state that a request by a principal p to perform the action a on a resource r is authorised only if p belongs to a category c such that for some category below c (e.g., c itself) the action a is authorised on r, otherwise the request is denied. Other alternatives (e.g., the possibility of considering *undeterminate* as answer if there is not enough information to grant the request, which is quite natural when composing policies both in centralised and in distributed systems), will be discussed in the next section describing the distributed version of CBAC.

3.3 Distributed CBAC

We consider the same sets of entities as in $\mathcal{M}$ plus a set of situational identifiers will include identifiers for sites (or locations) which will be associated with resources or policies. For simplicity we will assume that $\mathcal{S}$ is just the set of locations that compose the distributed system. In other words, each $s \in \mathcal{S}$ identifies one of the components of the distributed system, seen as a federation. The sets $\mathcal{P}, \mathcal{C}, \mathcal{A}, \mathcal{R}$ include, respectively, the principals, categories, actions and resources in any of the sites of the system.

In addition to the above relations, we define a notion of forbidden operation (or banned action) on resources, modelled by the relation $\mathcal{BARCA}$, and a notion of non-authorised access, modelled by the relation $\mathcal{BAR}$:

- *Banned actions on resources:* $\mathcal{BARCA} \subseteq \mathcal{A} \times \mathcal{R} \times \mathcal{C}$, such that $(a, r, c) \in \mathcal{BARCA}$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ is forbidden for principals assigned to the category $c \in \mathcal{C}$.
- *Banned access:* $\mathcal{BAR} \subseteq \mathcal{P} \times A \times \mathcal{R}$, such that $(p, a, r) \in \mathcal{BAR}$ iff performing the action $a \in \mathcal{A}$ on the resource $r \in \mathcal{R}$ is forbidden for the principal $p \in \mathcal{P}$.

Additionally, a relation $\mathcal{UNDET} \subseteq \mathcal{P} \times A \times \mathcal{R}$ could be defined if $\mathcal{PAR}$ and $\mathcal{BAR}$ are not complete, i.e., if there are access requests that are neither authorised nor denied (thus producing an undeterminate answer).

To take into account the fact that the system may be composed of several sites, with different policies in place at each site, we consider families of relations $\mathcal{PCA}_s$, $\mathcal{ARCA}_s$, $\mathcal{BARCA}_s$, $\mathcal{BAR}_s$, $\mathcal{UNDET}_s$ and $\mathcal{PAR}_s$ indexed by site identifiers. Intuitively, $\mathcal{PAR}_s$ (resp. $\mathcal{BAR}_s$) denotes the authorisations (resp. prohibitions) that are valid in the site s. The relation $\mathcal{PAR}$ defining the global authorisation policy will be obtained by composing the local policies defined by the relations $\mathcal{PAR}_s$ and $\mathcal{BAR}_s$ as indicated in the next section.

Definition 3.4 (Distributed Axioms) *In a distributed environment, the category-based metamodel is defined by the following core axiom where we assume that there exists a reflexive-transitive relation $\subseteq$ between categories; this can simply be equality, set inclusion (i.e., the set*

of principals assigned to $c \in \mathcal{C}$ is a subset of the set of principals assigned to $c' \in \mathcal{C}$), or an application specific relation may be used.

$$(b1) \quad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall s \in \mathcal{S}$$
$$((\exists c \in \mathcal{C}, \exists c' \in \mathcal{C}, (p,c) \in \mathcal{PCA}_s \wedge c \subseteq c' \ \wedge (a,r,c') \in \mathcal{ARCA}_s) \Leftrightarrow (p,a,r) \in \mathcal{PAR}_s)$$

Concerning the relation $\mathcal{BARCA}$ in a site s, the following axioms should be included:

$$(c1) \quad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall s \in \mathcal{S}$$
$$((\exists c \in \mathcal{C}, \exists c' \in \mathcal{C}, (p,c) \in \mathcal{PCA}_s \wedge c' \subseteq c \ \wedge (a,r,c') \in \mathcal{BARCA}_s) \Leftrightarrow (p,a,r) \in \mathcal{BAR}_s)$$

$$(d1) \quad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall s \in \mathcal{S}$$
$$((p,a,r) \notin \mathcal{PAR}_s \ \wedge \ (p,a,r) \notin \mathcal{BAR}_s) \Leftrightarrow (p,a,r) \in \mathcal{UNDET}_s$$

$$(e1) \quad \forall s \in \mathcal{S}, \quad \mathcal{PAR}_s \cap \mathcal{BAR}_s = \emptyset$$

Notice that access rights are inherited upwards through the hierarchy defined by the $\subseteq$ relation, while prohibitions are propagated downwards to the basis of the hierarchy.

Finally, the axioms below describe the global authorisation relation, which is obtained from the local authorisations and prohibitions defined at each site, by using operators $\mathcal{OP}_{par}$ and $\mathcal{OP}_{bar}$.

$$(f1) \quad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},$$
$$(p,a,r) \in \mathcal{OP}_{par}(\{\mathcal{PAR}_s, \mathcal{BAR}_s \mid s \in S\}) \Leftrightarrow (p,a,r) \in \mathcal{PAR}$$

$$(g1) \quad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},$$
$$(p,a,r) \in \mathcal{OP}_{bar}(\{\mathcal{PAR}_s, \mathcal{BAR}_s \mid s \in S\}) \Leftrightarrow (p,a,r) \in \mathcal{BAR}$$

$$(h1) \quad \mathcal{PAR} \cap \mathcal{BAR} = \emptyset$$

According to these axioms, the result of an access request may be different depending on the site where the request is evaluated, since each site has its own authorisation policy defined by the local relations $\mathcal{PAR}_s$ and $\mathcal{BAR}_s$ (see axioms $(b1)$ and $(c1)$). The relation $\mathcal{UNDET}_s \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$

is such that $(p, a, r) \in \mathcal{UNDET}_s$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ is neither allowed nor forbidden for the principal $p \in \mathcal{P}$ at site $s \in \mathcal{S}$ (see axiom $(d1)$); this implies that every tuple in $\mathcal{P} \times \mathcal{A} \times \mathcal{R}$ is in $\mathcal{PAR}_s \cup \mathcal{BAR}_s \cup \mathcal{UNDET}_s$. The axioms $(e1)$ and $(h1)$ preclude inconsistent specifications (i.e., a request cannot be both authorised and forbidden).

The final authorisation is computed by specialising the definition of the operators $\mathcal{OP}_{par}$ and $\mathcal{OP}_{bar}$, according to the application requirements (axioms $(f1)$ and $(g1)$). These operators take as parameter a set of local answers and combine them in order to produce a final access decision. They can give priority to positive authorisations, or to undeterminate (or negative) ones in case of conflict. While most of existing policy languages (e.g. XACML) have a fixed set of combination algorithms, our metamodel can be used to specify a large range of composition operators, as described in the next section.

3.3.1 The CBAC distributed semantics

An important aspect of the distributed metamodel is the capability of representing systems where resources may be dispersed across different sites, and the information needed to decide whether a user request is granted or denied may also be distributed. An overview of the rules modeling the distributed operational semantics of $CBAC$ is given in Table 3.1, where barca_s^* generalises the previously mentioned function barca_s to take into account lists of categories instead of a single category. As already mentioned, the authorization and prohibition sets should satisfy axiom $(e1)$, that is, $\mathsf{arca}_s^*(\mathsf{below}(\mathsf{pca}_s(p))) \cap \mathsf{barca}_s^*(\mathsf{above}(\mathsf{pca}_s(p))) = \emptyset$, but note that even if axiom $(e1)$ is not satisfied, the operational semantics defined for par above is consistent: it gives priority to authorizations.

The axioms $(f1)$ and $(g1)$ are realized by the rewrite rules $(f2), (g2)$ (implementing $\mathcal{OP}_{par}$ and $\mathcal{OP}_{bar}$ through the use of par_s; see rule $(c2, d2)$. The axioms $(f1)$ and $(g1)$ can be implemented in several ways. The version chosen in the definition above corresponds to a very general rewrite rule that can be used for evaluating an access request in a single central site, if $n = 1$ and the operator op is the identity, as well as for evaluating combinations of answers (with a suitable operator op) from n different local policies. An alternative can be to specify an $\mathsf{authorised}$

Par_s	$\mathsf{par}(p,a,r)$	$\rightarrow$ *if* $(a,r) \in \mathsf{arca}^*(\mathsf{below}(\mathsf{pca}(p)))$ *then* grant *else if* $(a,r) \in \mathsf{barca}^*(\mathsf{above}(\mathsf{pca}(p)))$ *then* deny *else* undeterminate	
$Arca^*$	$\mathsf{arca}^*(\mathsf{cons}(c,l))$	$\rightarrow \mathsf{append}(\mathsf{arca}(c), \mathsf{arca}^*(l))$	
$Arca^*$	$\mathsf{arca}^*(\mathsf{nil})$	$\rightarrow \mathsf{nil}$	
$Barca^*$	$\mathsf{barca}^*(\mathsf{cons}(c,l))$	$\rightarrow \mathsf{append}(\mathsf{barca}(c), \mathsf{barca}^*(l))$	
$Barca^*$	$\mathsf{barca}^*(\mathsf{nil})$	$\rightarrow \mathsf{nil}$	
Pca	$\mathsf{pca}(\mathsf{p})$	$\rightarrow [\mathsf{c}_1, \ldots, \mathsf{c}_k]$	*(for each principal p)*
$Arca$	$\mathsf{arca}(\mathsf{c})$	$\rightarrow [(\mathsf{a}_1, \mathsf{r}_1), \ldots, (\mathsf{a}_k, \mathsf{r}_k)]$	*(for each category c)*
$Barca$	$\mathsf{barca}(\mathsf{c})$	$\rightarrow [(\mathsf{a}_l, \mathsf{r}_l), \ldots, (\mathsf{a}_t, \mathsf{r}_t)]$	*(for each category c)*
$Above$	$\mathsf{above}([\mathsf{c}])$	$\rightarrow [\mathsf{c}, \mathsf{c}_1, \ldots, \mathsf{c}_n]$	*(for each category c)*
$Below$	$\mathsf{below}([\mathsf{c}])$	$\rightarrow [\mathsf{c}, \mathsf{c}'_1, \ldots, \mathsf{c}'_m]$	*(for each category c)*
Aut	$\mathsf{authorised}(p,a,r,s_1,\ldots,s_n)$	$\rightarrow \mathsf{fauth}(\mathsf{op}, \mathsf{par}_{s_1}(p,a,r), \ldots, \mathsf{par}_{s_n}(p,a,r))$	
$Fauth$	$\mathsf{fauth}(\mathsf{op}, \mathsf{answ}_1, \ldots, \mathsf{answ}_n)$	$\rightarrow \mathsf{answ}_0$	(where $\mathsf{answ}_i \in \mathcal{A}\mathsf{uth}$)

Table 3.1: Rewrite Specification of the Distributed Metamodel: Generic Functions, Specific Functions, and Combination Rules

rewrite rule for any specific combination operator. For example, if we consider two sites s_1 and s_2, for the precedence operator previously mentioned, we may have

$$\begin{aligned}\mathsf{authorised}(p,a,r,s_1,s_2) \;\rightarrow\; & \textit{if } \mathsf{par}_{s_1}(p,a,r) = \mathsf{grant} \textit{ or } \mathsf{par}_{s_1}(p,a,r) = \mathsf{deny} \\ & \textit{then } \mathsf{par}_{s_1}(p,a,r) \textit{ else } \mathsf{par}_{s_2}(p,a,r)\end{aligned}$$

In this case priority is given to local evaluation in site s_1, evaluation in site s_2 being executed only when the local policy in s_1 is not able to give as answer grant or deny. However, when dealing with policy combinations, it is unlikely to find a unique evaluation strategy that works for every possible scenario. A suitable policy integration mechanism depends on the requirements of the application and the involved parties. Evaluation initially takes place in the site where the request is issued, using the local function authorised (see rule *Aut* in Table 3.1). The local rule *Aut* specifies a number of sites s_i, $i = 1 \ldots n$, where the request may be passed and evaluated by the corresponding function par_{s_i}. For defining the sites of the evaluation, we may use functions like e.g. $\mathsf{psite}(p)$, which returns the site where the principal p is registered, or $\mathsf{rsite}(r)$ which returns the site where the resource r is located. In this way, access requests can be evaluated in a predefined central site, or priority can be given to local evaluation, or more elaborated

combinations of access answers can be implemented.

In order to specify a particular policy at a site s, e.g. RBAC, MAC or DEBAC, it is sufficient to specialize locally the functions arca_s, barca_s, pca_s, above_s, below_s. Their definition in Table 3.1 can be adapted to the application we want to consider. Thus, for example, to express a hierarchical RBAC policy at site s, the function arca_s will return the permissions associated to each defined role and below_s will return the roles inferior to a given role, according to the hierarchy specified in the model (see [BF10a] for more application examples). In the distributed metamodel, the request can be passed to other sites (with possibly different local policies) and evaluated in a distributed way. The generic rule (Aut), implementing the axioms defined in 3.4, is then used to integrate the different local access request answers to provide a final authorization decision. More precisely, the distributed answers are collected and combined by using specific rules $(Fauth)$, for a suitable operator op. We can specialize the combination rules $(Aut, Fauth)$ using a variety of algebraic operators.

3.3.2 Policy combining operators

Standard operators to compose policies, such as *Intersection*, *Subtraction* and *Union* can be easily defined in the metamodel. We briefly give next some intuition on the semantics of the above operators, applied to sets of authorisation answers.

- Using the *Intersection* operator (I), access is authorised (denied) if it is allowed (denied) by each of the component policies.
- Using the *Subtraction* operator ($-$), access is authorised (denied) if it is allowed (denied) by the first policy, but not by the second one.
- Finally, using the *Union* operator (U) an access can be authorised if it is allowed by any of the component policies. Since we have explicit definition of prohibitions as well as authorisations, conflicts may arise. We consider three union operators, depending on the conflict-resolution method used: U^G (i.e. access can be authorised if it is allowed by any of

the component policies), U^D (i.e. access is denied if it is denied by any of the component policies) and U^U (giving as answer undeterminate in case of conflicting information).

Moreover, the combination algorithms of XACML policy language [OAS03, LWQ+09], such as *Permit-overrides, Deny-overrides, First-applicable, Only-one-applicable* can be specified in our framework using recursive rules (for a complete specification see [BF08b]). One can note that in the setting of a three-valued policy (as we consider here) the XACML *Undeterminate* and *NotApplicable* results are treated in an equivalent way. We can easily obtain a four-valued policy by enlarging the set of answers $\mathcal{A}uth$ and by modifying the definition of the specific (*Fauth*) rule accordingly.

To facilitate the declarative definition of more sophisticated operators, it is useful to add higher-order features to the specification language. We describe next a higher-order extension of the language along the lines of [BF08b], in order to include policy combination operators such as override, closure and scope restriction (as defined e.g. in [BdCdVS00, WJ03a]) and further policy combination expressions including such operators. Restriction constrains the application of a policy to the requests satisfying certain conditions. Templates are used to define partially specified policies, that can be completed by supplying the missing parameters. Override replaces a part of a policy with a fragment of a second policy. The portion to be replaced is specified using a third policy.

The override operator can be expressed using a combination of the *Union* (U), *Substraction* ($-$) and *Intersection* (I) operators, obtaining an expression of the form $(a_1 - a_3)\mathsf{U}^{\mathsf{G}}(a_2 \mathsf{I} a_3)$, where a_i is the access control answer returned by the policy i (see [BF08b] for more details). We are able to model the expression above, and any other expression mixing the mentioned operators, by introducing in the specification language a (higher-order) function f which encodes the structure of the policy combinator we want to specify and contains variables that will be instantiated by the actual policy parameters at run-time (by β-reduction).

Formally, we introduce in the metamodel a rule that we call *Ho-Auth*:

Ho-Auth	$\mathsf{hoAuth}(f, p, a, r, s_1, \ldots, s_n) \quad \rightarrow \quad f \ s_1 \ \ldots \ s_n \ p \ a \ r$

where the variable f will be instantiated by a λ-abstraction of the form $\lambda\overline{s}'p'a'r'.e$: the bound variables $\overline{s}'$ represent the sites $s_1, \ldots, s_n$ involved in the evaluation, p', a', r' are variables to be instantiated by the actual parameters of the access request, that is the principal, the action and the resource, and e is a term specifying how the local policies are combined (in order to evaluate correctly, the term that instantiates f should not have free variables, that is, only $\overline{s}', p', a', r'$ may occur free in e, and when applied to the actual parameters of the access request it must have as normal form an authorisation answer in $\mathcal{A}uth$).

This additional expressive power allows us to generalise the Aut rule. We recall that in a completely distributed system, every local policy specifies its own Aut rule as well as the way combinations with external policies have to be performed. This means that, for a site s, we may have

$$\begin{array}{rcl} \mathsf{authorised}(p, a, r, s_1, \ldots, s_n) & \rightarrow & \mathsf{fauth}(\mathsf{op}, \mathsf{par}_{s_1}(p, a, r), \ldots, \mathsf{par}_{s_n}(p, a, r)) \\ & \mathit{or} & \\ & \rightarrow & \mathsf{hoAuth}(\mathsf{f}, p, a, r, s_1, \ldots, s_n) \end{array}$$

For example, to specify the override operator for the policies local to sites s_1, s_2, s_3, we use $\mathsf{f} = \lambda s_1 s_2 s_3 p' a' r'.e$ where $e = \mathsf{fauth}(\mathsf{U}^{\mathsf{G}}, e1, e2)$ with $e1 = \mathsf{fauth}(-, \mathsf{par}_{s_1}(p', a', r'), \mathsf{par}_{s_3}(p', a', r'))$, and $e2 = (\mathsf{fauth}(\mathsf{I}, \mathsf{par}_{s_2}(p', a', r'), \mathsf{par}_{s_3}(p', a', r'))$.

More detailed examples can be found in [BF08b].

3.4 Rewriting-based analysis of CBAC policies

Specifying access control policies via term rewriting systems, which have a formal semantics, has the advantage that this representation admits the possibility of proving properties of policies, and this is essential for policy acceptability [oD85]. Rewriting properties, such as confluence (which implies the unicity of normal forms) and termination (which implies the existence of normal forms for all terms), may be used to demonstrate satisfaction of essential properties of policies, such as consistency. More specifically, we are interested in the following properties of

access control policies.

Totality : Each access request from a principal p to perform an action a on a resource r receives an answer (e.g., grant, deny, undeterminate).

Consistency : For each access request from a principal p to perform an action a on a resource r at most one result can be obtained.

Soundness and Completeness : For any $p \in \mathcal{P}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, an access request by p to perform the action a on r is granted if and only if p belongs to a category c such that $c \subseteq c'$ and c' has the permission (a, r).

Totality and consistency can be proved, for policies defined as rewrite systems, by checking that the rewrite relation generated by the rules used in a specific instance of the metamodel is confluent and terminating. Termination ensures that all access requests produce a result (e.g. a result that is not grant or deny is interpreted as undeterminate) and confluence ensures that this result is unique. The soundness and completeness of a policy can be checked by analysing the normal forms of access requests. Sufficient completeness of the rewrite rules (a property that ensures that each operation is defined on all valid inputs) guarantees that the normal forms are of the right form [HCM05]. In practice, completeness can be easily achieved by assuming that either an open or closed policy operates as a default, and accordingly access is granted or denied if no authorization is found for it. Note that the alternative of explicitly requiring completeness of the authorizations is too heavy and complicates administration.

Confluence and termination of rewriting are undecidable properties in general, but there are several sufficient conditions for these properties to hold. We define next a condition that will be used to prove that policy specifications defined as first-order instances of the metamodel satisfy the properties we are interested in.

Definition 3.5 (Safe system) *A rewrite system R is safe if*

1. *R is non-overlapping;*

2. *R is a constructor system;*

3. *if a rewrite rule in R defines a recursive function* f*, then the right-hand side is built out of variables, constructors, previously defined functions, or recursive calls to* f *on arguments which are smaller (with respect to a well founded ordering, e.g. subterm) than the ones in its left-hand side.*

Conditions 1. and 2. in the definition of safe systems are natural: they require functions to be defined by cases on data constructors that do not overlap. They are standard conditions in functional programs. Condition 3. will be used to ensure termination [FJ95]. In [BF14] we show that to ensure that a policy specification obtained as an instance of the metamodel is consistent and total it is *sufficient* to check that the rules defining specific functions satisfy the safeness conditions.

Theorem 3.1 *Assume the rewrite system R defines an access control policy as an instance of the metamodel, using the rules in Table 3.1 with additional, specific rules satisfying the safeness conditions. Then, R is terminating and confluent.*

Proof 3.2 *Termination: First, observe that using the metamodel the full definition of the policy can be seen as a* hierarchical *rewrite system, where the basis includes the set of constants identifying the main entities in the model (e.g., principals, categories, etc.) as well as the set of auxiliary basic data (such as Booleans and numbers) and functions on them (*if-then-else, and*). The next level in the hierarchy includes all the auxiliary functions on lists (such as* append*) and the parameter functions of the model, that is, the specific functions* pca, arca, barca, above, below, $\in$. *The functions* arca^*, barca^*, *and* fauth *form the next level, followed by the definition of the function* par. *Finally the last level of the hierarchy consists of the definition of the function* authorised.

Several sufficient conditions for termination of rewrite systems defined as a hierarchical union of rules are available. For instance, we can use the following modularity result: a hierarchical

term rewriting system is terminating if the basis of the hierarchy is terminating and non-duplicating (i.e., rules do not duplicate variables in the right-hand side) and in the next levels of the hierarchy the recursive functions are defined, using previously defined functions, by rules that satisfy a general scheme of recursion, where recursive calls on the right-hand sides of rules are made on subterms of the left-hand side and there are no mutually recursive functions [FJ95].

The rules given in Table 3.1 satisfy the required conditions (notice that the functions par *and* authorised *are not recursive), and the assumption of safeness ensures the latter condition is satisfied for the additional rules used in a specific instance of the metamodel. We conclude that the system is terminating.*

To prove confluence, first note that the rules defining generic functions in Table 3.1 are non-overlapping, and the same holds for specific functions by the safeness assumption. Moreover, since the sets of generic and specific functions are disjoint, and the patterns are constructor terms, there are no critical pairs in the system, and therefore the system is locally confluent. Termination and local confluence imply confluence, by Newman's Lemma [New42].

As a consequence, every term has a unique normal form and this implies that the policy specification is *consistent*.

Property 3.3 (Consistency) *Assume R defines an access control policy as an instance of the metamodel, using the generic rules in Table 3.1 and additional, specific rules satisfying the safeness conditions. It is not possible to derive, from R, both* grant *and* deny *for a request* $\mathsf{authorised}(\mathsf{p}, \mathsf{a}, \mathsf{r}, \mathsf{s}_1, \ldots, \mathsf{s}_n)$.

We give next a precise characterization of the normal forms of access requests.

Proposition 3.4 *Assuming the specific functions used in an instance of the metamodel (specific versions of* pca, arca, barca, above, below, fauth*) are well-defined (i.e., their evaluation produces a result provided the arguments are valid; for instance, the evaluation of a ground term* $\mathsf{pca}(\mathsf{p})$ *results in a list of categories for any principal* p, $\mathsf{above}(\mathsf{c})$ *and* $\mathsf{below}(\mathsf{c})$ *produce lists of categories*

for any category c, $\mathsf{arca}(\mathsf{c})$ *and* $\mathsf{barca}(\mathsf{c})$ *produce lists of pairs* (a, r), fauth *produces a result in* $\mathcal{A}uth$*), then the normal form of a ground term* $\mathsf{authorised}(\mathsf{p}, \mathsf{a}, \mathsf{r}, \mathsf{s}_1, \ldots, \mathsf{s}_\mathsf{n})$ *where* $\mathsf{p} \in \mathcal{P}$, $\mathsf{a} \in \mathcal{A}$, $\mathsf{r} \in \mathcal{R}$, $\mathsf{s}_1, \ldots, \mathsf{s}_\mathsf{n} \in \mathcal{S}$, *is in* $\mathcal{A}uth$.

As a consequence, our specification of the access control policy is *total*.

Property 3.5 (Totality) *Assuming the specific functions used in an instance of the meta-model (specific versions of* pca, arca, barca, above, below, fauth*) satisfy the safeness condition and are well-defined (i.e., the evaluation of a ground term* $\mathsf{pca}(\mathsf{p})$ *results in a category for any principal* p*, and similarly for the other functions, as described in Prop. 3.4), each request* $\mathsf{authorised}(\mathsf{p}, \mathsf{a}, \mathsf{r}, \mathsf{s}_1, \ldots, \mathsf{s}_\mathsf{n})$ *receives an answer in* $\mathcal{A}uth$.

Similar results are shown for the higher-order version of CBAC in [BF14].

Liveness and Effectiveness. A policy is ineffective if there are principals who have not got any permissions or resources which are not accessible by any principal. These properties can be analysed for CBAC policies by checking the definitions of the categories of principals and resources and the relations pca and arca, as follows.

To check that all principals have authorisation to perform at least one operation on a resource r, it is sufficient to check that for every principal p there is at least one tuple in the relation pca of the form (p, c) such that c occurs in at least one tuple (a, r, c) of the relation arca,.

To check that all resources are accessible by at least one principal, it is sufficient to check that for each resource r there is at least one tuple in the relation arca of the form (a, r, c) such that c occurs in at least one tuple of the relation pca.

Both of these checks are polynomial in time on the size of the policy and can be implemented after each administrative operation to ensure updates do not produce ineffective policies.

To check the property of liveness (which requires each resource to be protected by the policy), assuming we have a list of the actions on resources that must be protected, we can simply check

that each resource r and for each action a that must be protected on r there is at least one tuple (a, r, c) in the relation arca. Again, this is a polynomial time check that can be implemented after each administrative operation.

Safety. The property of safety was initially stated for operating systems [HRU76a], to establish whether or not a given user can obtain a certain right on a certain object, and it was shown to be undecidable in the general case. Safety was redefined for RBAC systems as checking whether there exists a sequence of administrative operations that would permit a certain user to acquire certain roles, even if currently they do not have those roles, and it was shown to be decidable by reducing it to a reachability problem over a finite graph. For dynamic access control models such as ABAC, the safety property boils down to checking whether a user can acquire attribute values that grant access to a resource. For the particular case of $ABAC_\alpha$ [JKS12a] (where attributes range over finite domains, there is at most one authorisation rule that can apply to each permission, and only administrators can modify values of attributes) safety has also been shown to be decidable (see [AS17, MKD19]).

In the case of CBAC, the safety property is not decidable in general, however, we can characterise classes of policies for which safety is decidable. Below we show the undecidability of safety in general, before defining decidable subclasses of CBAC policies.

Theorem 3.6 (Undecidability of CBAC Safety) *The safety problem for CBAC policies is undecidable in general.*

Proof 3.7 *First, notice that in the CBAC model category conditions can be arbitrary formulas, with the only requirement that satisfiability should be decidable (to ensure that the CBAC relationships are computable). To solve the safety question, we need to determine if there exists a combination of operations that may lead to a user acquiring a specific permission (a, r). This implies that we need to determine if one or more users $u_1, \ldots, u_n$ could perform a sequence of operations that lead to a user belonging to a category which has the permission (a, r).*

To prove undecidability of safety, we will show that the Halting Problem can be reduced to the safety problem. More precisely, we will show that for any Turing machine M and input I, there exists a CBAC policy that simulates the execution of M on I, such that M halts on I if and only if the CBAC policy authorises the user u_s to perform the action a_s on r_s. The CBAC policy that we use to encode the Halting Problem as a safety problem includes a resource r_M representing M, and two actions read and next, such that the read action inputs I (the input for the Turing machine M), computes the initial configuration of M on I and stores it in r_M, and the next action computes the next configuration of the machine (i.e., it implements the transition function of the Turing machine M). In addition, we define a category c_0 such that $(read, r_M, c_0) \in \mathsf{arca}$ and $(next, r_M, c_0) \in \mathsf{arca}$, and such that every user belongs to c_0. In this way, according to the axiom $(a1)$, any user is authorised to execute the read and next actions on r_M. We also define a category c_F such that $(a_s, r_s, c_F) \in \mathsf{arca}$, and specify that $(u_s, c_F) \in \mathsf{pca}$ if and only if the current configuration of the machine, stored in r_M, is a final configuration (this is a decidable test). Now, according to $(a1)$, $(u_s, a_s, r_s) \in \mathsf{par}$ if and only if M reaches a final configuration, or equivalently, u_s obtains the permission to execute a_s on r_s if and only if M halts on I.

RBAC policies and $ABAC_\alpha$ policies can be seen as particular classes of CBAC policies, so using previous results on decidability of safety for these models, we can directly deduce decidability for the corresponding CBAC classes. However, it is possible to extend the result to cover CBAC policies that are not RBAC or $ABAC_\alpha$, where the reachability problem associated to safety is still bounded. Below we specify such class, which we call *Finitary CBAC*, by defining an operational semantics for policies in the form of a transition system.

Recall that a CBAC *policy* is a tuple $\langle E, Rel\rangle$ that defines the contents of the family of entities $\mathcal{E}$ and relations $\mathcal{R}el$, i.e., in the tuple there is a component for each set of CBAC entities and each CBAC relationship, which permits to compute the answer to any access request via the CBAC axioms.

Definition 3.6 (Finitary CBAC Policies) *A CBAC policy transition, written $\langle E, Rel\rangle \rightarrow \langle E', Rel'\rangle$, is a pair of policy tuples, such that there exists an action that is authorised in $\langle E, Rel\rangle$*

and whose execution results in $\langle E', Rel' \rangle$.

A CBAC policy $\langle E, Rel \rangle$ *is finitary if there is only a finite number of different policy tuples that are reachable from* $\langle E, Rel \rangle$ *via policy transitions.*

Finitary CBAC policies can be seen as mapping to a finite rewrite relation, for which reachability is trivially decidable, and as a consequence safety is decidable.

Theorem 3.8 (Decidability of Safety for Finitary CBAC policies) *Finitary CBAC policies have a decidable safety problem.*

Proof 3.9 *An easy way to decide safety is to generate the full transition relation for the given policy (i.e., compute the full set of tuples that can be obtained by performing transitions from the initial policy) and search for a tuple that grants the specific permission sought. Then, the sequence of transitions from the initial policy to the one that grants the permission indicates the sequence of operations that needs to be performed for the user to acquire the required permission. Since the set of policies that can be reached from the initial one is finite by assumption, this is a decision procedure.*

Constraints. In many practical cases, it may be useful to avoid for a user the possibility of executing conflicting actions (SoD, or Separation of Duties constraint) or, on the contrary, to ensure the possibility for the same user to execute two related actions (BoD, or Binding of Duty constraint). These properties can be expressed for CBAC policies at the metamodel level, by adding the appropriate axioms. For example, given two conflicting (respectively, related) actions a_1 and a_2 we may specify

$$
\begin{aligned}
(SoD) \quad & \forall p \in \mathcal{P},\ \forall c_p \in \mathcal{C},\ \forall r, r' \in \mathcal{R}, \\
& ((p, c_p) \in \mathcal{PCA}) \wedge ((a_1, r, c_p) \in \mathcal{ARCA}) \Rightarrow \\
& \nexists c' \in \mathcal{C}, s.t.\ (a_2, r', c') \in \mathcal{ARCA} \wedge (p, c') \in \mathcal{PCA})
\end{aligned}
$$

$$
\begin{aligned}
(BoD) \quad & \forall p \in \mathcal{P},\ \forall c_p \in \mathcal{C},\ \forall r, r' \in \mathcal{R}, \\
& ((p, c_p) \in \mathcal{PCA}) \wedge ((a_1, r, c_p) \in \mathcal{ARCA}) \Rightarrow \\
& \exists c' \in \mathcal{C}, s.t.\ (a_2, r', c') \in \mathcal{ARCA} \wedge (p, c') \in \mathcal{PCA})
\end{aligned}
$$

Both of these constraint verifications can be done in polynomial time on the size of the policy. Note that this implements a strong version of the constraints. Indeed, even if (SoD) or (BoD) are not satisfied by design, there could be executions that do not violate the principles of binding or separation of duties, e.g., in (SoD) the category c' exists but it is not used by the user in the current execution.

3.4.1 Policy administration analysis by narrowing

Rewriting techniques can also be used to study administrative actions implications on a rewrite-based policy evolving over time. The administration of access control policies is very important and must be carefully controlled to ensure the initial policy does not drift away from its original objectives. The administration of the category meta-model implies several actions, as for instance users can be reassigned from one category to another, categories can be granted new permissions as new applications and systems are incorporated. Permissions can be revoked from categories as needed. Category hierarchies can be created according to the organizational setup and can be changed thereafter. A (sequence of) administrative action(s) may cause profound transformations in the behavior of the policy. A possible solution would be to formally verify again all properties of the policy after each modification. However, this may be a long and expensive task and, moreover, in the case one of the desired properties is invalidated, often the administrator has no indication to solve the problem. Therefore, in order to facilitate the work of policy administrators during the evolution of a security policy, an alternative (and complementary) approach is to identify examples where the old and the new version of the policy behave differently. We consider two different cases:

- *Preservation*: the administrator modifies the implementation of the policy (for instance by reducing the set of rules defining it) without aiming at modifying the policy behavior,

- *Non-regression*: the administrator modifies the implementation of the policy (for instance by adding or removing users and/or permissions) without aiming at modifying the behavior of the policy on the initial set.

In [BTV16] we have developed a technique that we call *narrowing differential* to identify entities in a policy whose behavior has been (unintendedly) affected by administrative actions, i.e., saying it in the rewriting phrasing, to identify terms whose reduction changes in the two rewrite systems representing the two versions of the policy before and after the administrator's intervention.

We can use narrowing in order to query an access control policy expressed as a term rewrite system, in a more broad way than with simple rewriting [KKdO09]. Rewriting enabled us to resolve access requests encoded as ground terms. Narrowing enables to perform requests encoded as terms with variables and to enumerate all the solutions for those requests. We can also easily encode a request of the system administrator as an equation to resolve by narrowing.

We focus on investigating how a modification of the policy reflects on its behavior and how to represent this behavioral change. In rewriting terms, this issue can be expressed as follows: given two rewrite systems R_o and R_n (representing the old and the new version of the policy) we want to identify the (common) terms leading to different derivations in R_n and R_o. Our approach is developed to deal with the two cases mentioned above, that is

- we have developed the technique of narrowing differential for rewriting systems defined on the same set of constructors and
- we have proposed an extension of this technique for rewriting systems defined on two sets of non-disjoint constructors.

For example, let us consider the case where the policy has been re-implemented (e.g. in the aim of simplifying its set of rules). We want to check that the modifications have introduced no dangerous side effects on the behavior of the policy. To represent this, we consider two rewrite systems $\mathcal{R}_1$ and $\mathcal{R}_2$ with the same set of constructors and two non-disjoint sets of

operations. We verify the policy behavior by comparing the possible derivations in the two rewrite systems for a given query, denoted t. If a counter-example is detected, meaning that security may be compromised, this is exhibited in the set of differentials, denoted $\mathbf{D}_{\mathcal{R}_1,\mathcal{R}_2}(t)$. We can also consider different sets of constructors. Intuitively, from the security point of view, this represents an evolution of the policy where the set of categories, permissions or users have been changed and the related rules have been modified. In both cases, our technique is shown to be sound and complete.

We have provided an implementation of our technique for the class of inductively sequential constructor-based term-rewrite systems where the outermost needed narrowing strategy is used for computing all the constructor substitutions that are solution of the narrowing equation, i.e. for enumerating all the different policy behaviors.

3.5 Related work

Term rewriting has been used to model a variety of problems in security, from the analysis of security protocols (see, for example, [BDHdS02, EMM06, Vig05, Bla16]) to the definition of policies for controlling information leakage [EP05]. On access control specifically, Koch et al. [KMPP04] use graph transformation rules to formalise RBAC, and more recently, [BF06, San08, BFB07] use term rewrite rules to express access control policies. In [DKKdO07], rewriting reduction strategies are used for specifying policy composition. Other formal theories to define and validate security policies (see, for instance, [BS03]) have used first-order theorem provers, purpose-built logics, or flow-analysis but these approaches have limitations (as discussed for instance in [JS05]).

Several Access control metamodels have been proposed in recent years, with advanced features for specifying and enforcing different access control policies. Similarly to our work, the model presented in [SKAL12] is also based on an extension of the metamodel of access control proposed by Barker [Bar09]. Authors developed UACML, an UML-based modeling framework for the visual design of hybrid access control policies. They also provide a textual language that

corresponds to UACML and present a Prolog-based verification tool for detecting errors within the textual language, so to ensure a correct specification in UACML by repeating verification and translation cycles. The work in [AA20] presents a higher-order generalisation of ABAC providing a hierarchical structure over attributes by means of aggregation operations. This additional level of abstraction w.r.t. the standard ABAC model is similar to the notion of categorization and category hierarchy that we have defined. Most recent access control metamodels are tailored to address the problem of data protection in the new networking environments, such as IoT applications, Cloud computing, or web content and services management [KAAI21]. However, most of these works are specified using the UML language [KAI21, KM16], which lacks a clear formal semantics, or implementation oriented, e.g. by extending the Spring Security framework [GNS17].

Concerning administrative actions and the impact of their application on access control policies, a number of researchers investigated the problem, mainly in the setting of Administrative RBAC policies [LT06, FMP13, SYSR06]. In particular, a number of analysis techniques have been developed for user-role reachability analysis of ARBAC [SYRG07, JLT+08, AR12]. The reachability (or safety) problem asks whether a given user u is a member of a given role r in any policy reachable from the initial (i.e., current) policy by actions of a given set of administrators. Some other analysis problems, such as permission-role reachability, user-role availability, and role-role containment, can be solved by reducing them to the user-role reachability analysis problem [SYRG07].

Reachability over term rewriting is a very general approach which constructs approximations of the set of normal forms with respect to a rewrite system, see for instance [Gen98]. This is done through the use of tree automata. When verifying some security property, one tests for the reachability of terms representing unwanted situations. This approach has been followed on the development of static analyzers [BGJLR07]. Narrowing techniques have been used in [MT07] to solve reachability problems in cryptographic protocols verification. More close to our work, in [KKdO09] authors use strategic narrowing to solve queries in a rewrite-based policy specification and give examples of *what-if* analysis.

Several policy Datalog-based languages, such as SecPal [BFG10], Binder [DeT02], SD3 [Jim01], RT [LMW02] and Cassandra [BS04], have been proposed for distributed access control policies modeling. SecPal, for instance, is a simple yet very expressive language: it can express many common policy idioms using constraints, controlled delegation, recursive predicates and negated queries. All these declarative languages, being based on a monotonic semantics, are not especially well suited for representing dynamically changing distributed policies. Recent research addressed these limitations by extending Datalog to include notions such as updates and persistency, but these features have an operational semantics outside Datalog, which may cause semantic ambiguities and increases the complexity of the language and its interpretation, [NR09, Mao10]. Dedalus [AMC+11] is an extension of Datalog with an explicit notion of time for expressing the dynamics of distributed systems. However, dealing with time explicitly is not always necessary. The framework that we have described is more expressive than any of these Datalog-based languages, as it can include higher-order features.

Among the languages for distributed access control, we can also also mention XACML [OAS13] as a general language for access control policy specification. However, we believe a general access control language should come with a well-defined access control model and a sound formal semantics. The language designed by Jajodia and colleagues in [JSSS01], known as *Flexible Authorization Framework*, FAF, tries to balance expressivity and performance. It allows users to specify, together with the authorizations, the policy according to which access control decisions are to be made. A set of conflict resolution operators is proposed, to disambiguate among conflicting policy decisions.

With respect to policy composition, a number of works have a close relationship with the formalization presented in this thesis. Policy composition is addressed in [BVS02] through an algebra of composition operators that is able to define union, intersection, policy templates, among other operations. The operator definitions can be adapted to several languages and situations, since their definition is orthogonal to the underlying authorization language. The work presented in [WJ03b] extended Bonatti et al.'s algebra with negative authorizations and non-determinism, features that our rewrite-based framework supports naturally . Another alternative for composing access control policies is implemented by the Polymer system [BLW05]

which supports the reuse of policies treated as first-class objects in the language. The formal semantics of Polymer is based on a variant of the lambda calculus and some higher-order features are supported similarly to our work (Policies may be modified with new actions to be executed when some security event is triggered, in a sort of higher-order composition).

Dougherty et al. [DFK06] present a model for formal analysis of access-control policies in their dynamic environments. In particular, they propose a new mathematical model of policies, their environments, and the interactions between them. Policy analysis is provided using a combination of relational reasoning and temporal reasoning. A unique feature of their model is that it separates the (static) policy from its (dynamic) environment and this allows for analysis applied to the static policy alone.

Elisa Bertino et al [BCFP03] propose a formal framework for reasoning about different access control models. Their framework is logic-based and can capture discretionary, mandatory, and role-based access control models. Each instance of the proposed framework corresponds to a C-Datalog program, interpreted according to the stable model semantics. The authors show that checking for structural subsumption/equivalence between different access control models is decidable, however access request equivalence is not.

There are also proposals for analyzing policies based on description [ZHLL05] or modal logics [Mas97]. Both of them provide a formalization of role based access control, and show how tableau-based decision methods can be used for consistency checking of policies, evaluating access requests and verifying policies against security properties.

Chapter 4

Controlling access in collaborating component systems

Although providing clear benefits to users, organizations and society, cooperation introduces new challenges in the governance of data and especially in the control of its usage. In fact, information gathered by cooperative systems is distributed, i.e. physically located on different hosts, and heterogeneous, i.e. coming from sensors and internal databases as well as from public sources like blogs and social media. Moreover, access control policies are typically managed independently and maintained by different organizations that might wish to retain control over their resources. The trust of parties, their willingness to collaborate and thus the added value of the collaborative system all depend on offering contributors appropriate control over the resources they share.

We present in this chapter our solutions to two particular problems.

The first concerns the specification of authorizations for inter-operable web services, as described in Section 4.1. In this context, a particularly difficult case occurs when a service invokes another service to satisfy an initial request since, even though the user had the rights to access the original service, he/she might not have the permissions to access the invoked services, leading to indirect authorization errors.

The second problem is related to data coming from multiple, heterogeneous information sources, which are processed and fused into new pieces of information that can be, in turn, further processed by other entities participating in the cooperation. Controlling the access to such evolving and variegated data, often under the authority of different entities, is challenging. The solution we propose is described in Section 4.2.

4.1 Service Oriented Architecture dependencies

Nowadays organizations increasingly employ distributed systems in order to improve their service performance. Web services, which are a form of distributed system architecture, seem to become the preferred implementation technology for realising the integration and interaction between various systems in Internet and Intranet environments. They also offer many benefits over other types of distributed computing architectures, such as maximum service sharing, reuse and interoperability. The reference paradigm as of now is the Service Oriented Architecture (SOA) [PG03], in which applications can be given a standard interface and stored as reusable units for composition. SOA does not require the integration of components and code into one computing environment; instead, it requires the specification of data exchanges so that the final result can be produced by exploiting the results of the cooperating services. SOA subsumes all the problems and issues of distributed computing such as non-determinism, communication, or synchronization, while adding a number of additional problems in particular related to security. Typically, SOA is based on web services standards which use XML as the communication format. The relevant standard for access control is the Security Assertion Markup Language (SAML) [Hug05] which provides a mean for exchanging security information across domain boundaries and can be combined with both SSO solutions for authentication and the eXtensible Access Control Markup Language (XACML) for authorization. While the SAML specification is independent of the kind of assertions that can be made, many instances include the specification of an identity, i.e. an entity authenticated within a given domain in the federation.

As a consequence of this and as explicitly suggested in [Hug05], most available implementations of access control enforcement mechanisms based on SOA relate access control decisions to the identity of the requester. On the one hand, this simplifies accountability since SAML assertions carry identity information that allow for tracking who is getting access to what. On the other hand, it complicates the authorization process since the identity information in the SAML assertion must be used to retrieve the attributes associated to the identity of the requester which are relevant to the evaluation of the authorization conditions in the access control policies. This may be very difficult in a federated system because of the authorization problem related to access dependencies.

Dependability refers to the property of a system by which some trust can be placed on the delivered service. Since SOA systems are usually obtained by composition of simpler services, dependability is the result of the capability of the component services to deliver certain results and the way these results are exchanged among the services. This aspect is particularly important when security properties come into the picture and authentication or authorization of users with respect to the integrated services must be enforced.

It is important to notice that specifying authorization conditions across different domains is significantly more complex than federating identities. To understand why this is the case, consider the situation in which a service A needs to invoke another service B to complete its computations. In this situation, even though a subject had the rights to access service A, it might not have the permissions to access service B. The problem is that, even if the services belong to the same federation for which the subject has been authenticated, its attributes are relevant to the evaluation of the access control policies in the domain of service A but they may be unknown to the domain of service B since the access control policies of the two services were designed autonomously and expressed in terms of different sets of attributes. This situation is an instance of the access control problem in presence of transitive dependencies, see e.g., [LK07].

To overcome this problem, a possibility is to give full access rights to services, but this violates the least privilege principle and allow to circumvent the intent of access policies, thereby paving the way to security exploits. In addition, when data from service A is stored in service B, it

may be possible for users to circumvent the intent of the access control policy of service A by reading data from service B. In general, this kind of errors are difficult to prevent through testing once the federated system has been deployed, because of the large number of possible execution scenarios from which one must select the few that give rise to unsecure situations.

4.1.1 Synthesis of our approach

In our work, we focus on a particular class of federated systems, called Enterprise Information Portals (EIPs). EIPs enable companies to unlock internally and externally stored information while providing users with a single access point to personalized information needed to make informed business decisions [ST98]. One hallmark of enterprise portals is the de-centralized content contribution and management, which keeps the information always updated. Another distinguishing characteristic is that they cater for customers, vendors, and others beyond an organization's boundaries. This contrasts with a corporate portal which is structured for roles within an organization. EIPs provide a secure unified access point, often in the form of a web-based user interface, and are designed to aggregate and personalize information through application-specific portlets. Roughly, an EIP allows for integrating information, people, and processes across organizational boundaries in a manner similar to the more general Web Portals. The notion of EIP is a widely adopted solution to the question of how to integrate and incorporate many different and possibly heterogeneous systems—which are generally independently designed and developed—while allowing for seamless access. EIPs are used in all kinds of companies, from multinationals to Small Enterprises, but also in public organizations such as governments, schools, and hospitals.

We focus on EIP where the topology of the system (i.e. how the integrated services may invoke each other) is known and fixed and we pay particular attention to the authorization problem arising from the presence of transitive dependencies. Our goal is to ensure that the global policy is consistent with those of the individual services (according to the principles of autonomy and security discussed above) and to avoid the authorization problem related to transitive

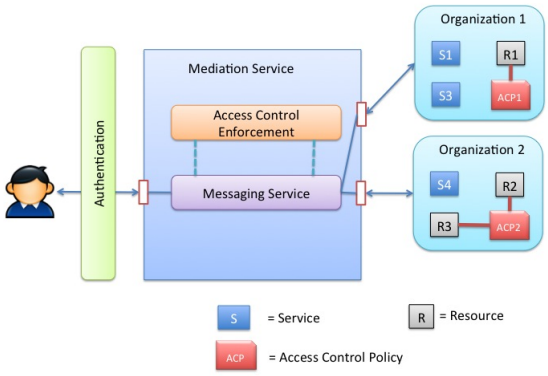


Figure 4.1: Access control with a mediation service

dependencies. Implementations of EIPs adopt the SOA paradigm by using web services;[1] we can thus check the practical applicability of our ideas and implement enforcement mechanisms following SOA best practices by building on top of available FIdM platforms.

We first develop a framework for the modeling of access control policies for EIPs. Following [Bro08], the EIP is the ideal place where to locate the policy enforcement point which, together with a module for messaging, provides the interface to both users and the different services protected by their own authorization policy. We call this module *mediation service* (see, Figure 4.1). The mediation service can enforce policies governing access to its own interfaces, since the services do not directly communicate with each other but they communicate via the interfaces provided by the mediation service. In this way, it can authenticate the component trying to gain access and check component's authorization to use the mediation service. It is the ideal place where to keep track of the selection of the appropriate policy to enforce as well as to keep track of the transitive dependencies.

To exemplify our approach, we consider a case study based on a research center that has developed an IT solution for authorization management of its researchers travel orders (hotel reservation, transportation, and other associated expenses when traveling to a scientific event). The research center is composed of 4 departments [Figure 4.2]:

- Secretary's Office (SEC) receives the requests from the researchers, makes an estimation

[1] G. Phier. "A Portal May Be Your First Step to Leverage SOA." Available at `https://www.gartner.com/doc/485862/portal-step-leverage-soa`.

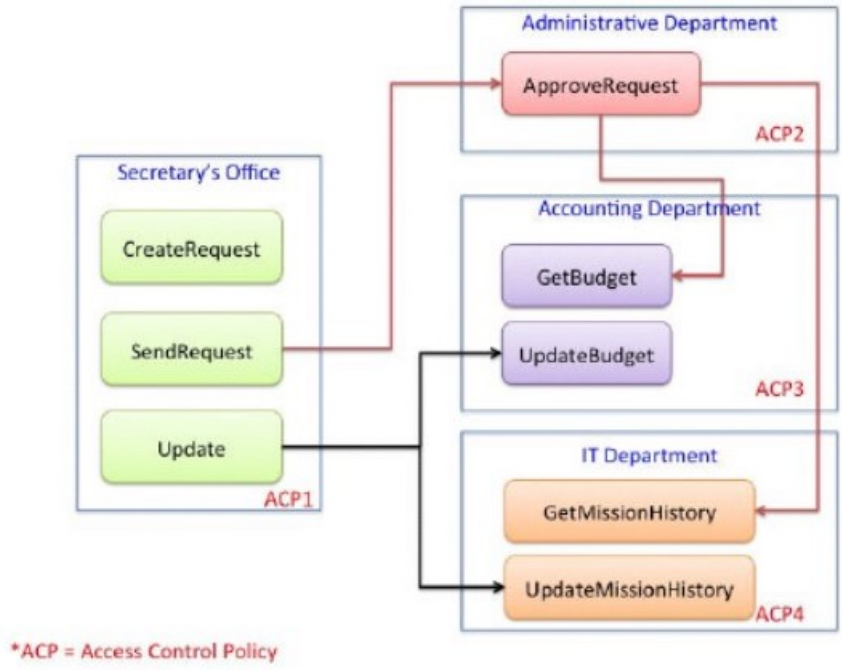


Figure 4.2: Research center travel authorization management schema

of the costs and creates the authorization request;

- Administrative Department (ADD) analyzes and validates the budget requested and approves the authorization request;
- Accounting Department (ACD) manages the research center's budget;
- IT Department (ITD) manages historical data of researchers' travels.

In this setting, a researcher (Subject) who wants to travel to a conference needs to contact the secretary for the accommodation and travel bookings and the payment of the conference registration fees. The secretary (Initiator) then creates a costs estimation and generates an authorization request, which is forwarded to the team leader or the person in charge of the demanded budget. Before validating an authorization request, the team Leader (Approver) must check if the researcher has not yet exceeded his quota of traveling by asking the IT department for his travel history, and check if the estimated costs are not too high by contacting the accounting department. Once the decision taken (validation or refusal), the answer is

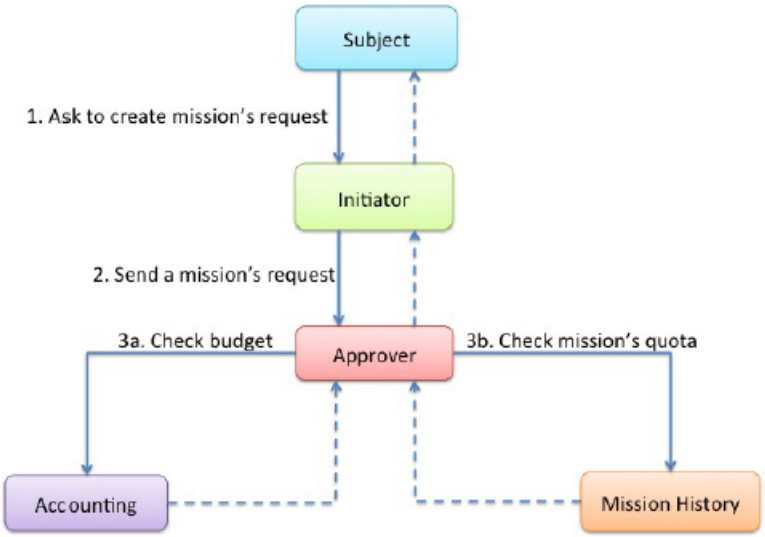


Figure 4.3: Research center travel authorization transitive calls schema

returned to the secretary for notification to the researcher and the updating of the system. This scenario is illustrated in Figure 4.3.

Technically, we develop our approach by extending the ORganization Based Access Control (OrBAC) model of [KBB+03]. OrBAC combines concepts like roles, temporal authorizations, obligations, and recommendations. It introduces a new dimension to policy specification (in addition to roles and groups) that is useful for the exchange of policies between different entities. The main concept of the model is the notion of *organization*, which provides scopes to policies. Each security policy is defined for and by an organization and the specification of the policies is parametrized by the organization so that it is possible to handle simultaneously several security policies associated with different organizations. The model is based on first-order logic and uses Datalog as specification language. Conflicts are solved by different approaches (see e.g. [ZZ08, CCBG07, Miè05]): one can adhere to conditions (or guidelines) which help to avoid conflicts, or priorities can be assigned to decisions.

Even if OrBAC improves the management of the security policy and reduces highly its complexity, it is partly limited since it is not adapted to distributed and interoperable systems. As security permission rules take a single referring organization as a parameter, it is not possible to represent rules that involve several organizations. Consequently, OrBAC is not well-adapted for

organization(s,org)	*subject s belongs to organisation org*
belong-to(sv, org)	*service sv belongs to organisation org*
permission(org,cat,act,o)	*category cat can do action act on object o in organization org*
empower(org,s,cat)	*subject s is assigned to category cat in organization org*
authorize(org,cat, act, o)	*category cat in organization org is allowed to do action act on object o*
is-permitted(s,act,o)	*subject s requests action act on object o*
delegate(org1, cat1, org2,cat2)	*category cat*1 *in organisation org is associated to category cat*2 *in organisation org*2
depends-on(sv1,sv2)	*service sv*1 *depends on (uses) service sv*2

Table 4.1: Main predicates in the D-OrBAC model

modeling a dynamic, decentralized environment where organizations cooperate while respecting the authority of each other over its own users and resources.

This Chapter is based on the work published during Worachet Uttha's PhD thesis in [UBR15, UBR14, Utt16].

4.1.2 Proposed model

As already mentioned, although the OrBAC model can deal with the specification of policies for organizations or sub-organizations, the transitive access request is still not solved. In distributed environments, a subject may be recognized in an organization but it may not be known outside it. We will introduce the delegation of authority in order to overcome this issue.

OrBAC originally structures subjects into roles as in RBAC: a subject is pre-assigned to a (set of) roles which in turn are associated with permissions. In order to increase the flexibility and meet interoperability between organizations, in our approach we structure the subjects in a general entity called category (as in chapter 3) and the role is considered as a particular instance of a category. Categories are associated with users on the base of the credentials they own, and permissions are assigned to each category.

In our extended OrBAC model we have the following main entities:

- **Organization and Services** : An *Organization* (denoted org_0, org_1, ...) can be seen as

an organized group of activities (in the research center, we may have the administrative department, the accounting department, etc.) or it can be seen as an organized group of services (i.e. a web portal). A service inside an organization is denoted by sv_0, sv_1, ...

- **Subjects and Categories** : The entity *Subject* (denoted s_0, s_1, ...) is an active entity, (i.e. a user or an organization). Subjects that have the same attributes' value or satisfy same conditions belong to the same group, called *Category* (denoted cat_0, cat_1, ...) and have the same permissions.

- **Objects and Actions**[2] : *Objects* (denoted o_0, ...) are passive entity such as data files, researcher's records, etc. *Action* entities (denoted act_0, act_1, ...) contain computer actions such as "read", "write", "send" or "print".

The access control policy is modelled using the predicate *Permission(org, cat, act, o)* that specifies the permissions between categories, actions and objects, while the evaluation of requests is modelled using the predicate *Is_permitted(s, act, o)* that allows to derive permissions between subjects, actions and objects. The Subject-Category assignment is modelled using the predicate *Empower(org, s, cat)* (see Table 4.1)

In the organization org, a subject s has a permission to perform an action act on an object o if (i) s is associated to a category cat in the organization org and (ii) the organization org grants the category cat the permission to perform the action act on the object o, then an request of the subject s to perform the action act on the object o is accepted. The derivation of permissions is modelled by the following rule:

$$\forall org, \forall cat, \forall s, \forall o, \forall act,$$

$$Permission(org, cat, act, o) \wedge Empower(org, s, cat) \leftarrow Is\_permitted(s, act, o) \quad (4.1)$$

Subject, action and object attributes are modelled by a set of binary predicates having the form *attribute_name(entity, value)*. We recall that the subject's attributes are meaningful in relation

[2]For easing the notation, we do not consider the abstraction of objects and actions as in the original OrBAC model, but this can be easily accommodated by adding 2 extra entities and the corresponding relations.

Figure 4.4: Delegation graph for the research center

with the organization he/she belongs to, and the same holds for the objects of the system. We can model the Subject-Category assignment according to the attributes of the subject as follows:

$$\forall org, \forall cat, \forall s,$$
$$Empower(org, s, cat) \leftarrow org\_attr_1(s, val_1) * org\_attr_2(s, val_2) * \cdots * org\_attr_n(s, val_n) \quad (4.2)$$

where * can be disjunction($\vee$), conjunction ($\wedge$) or a mixture of both.

Delegation or category mapping As our system is distributed across several organizations, each having a different authorization domain, we have defined a delegation policy specifying the mapping of categories belonging to different authorization domains. This is formalised by a delegation graph in the form of Directed Acyclic Graph (DAG) which describes the way a category may be delegated (see Fig. 4.4). Each node represents a service authorization domain which can delegate categories (and thus permissions) to another service authorization domain. This suppose that an agreement on a set of categories that are allowed to be delegated has been previously reached between the different participants.

In the presented research center scenario, we have restricted ourselves to the simple association

of categories (one-to-one) because of the difficulty in reaching a consensus of the association of categories (or attributes) between two or more domains. This problem is considered difficult because organizations may have categories of different names but with the same meaning or have categories with the same name but not the same meaning. In the related works on the association of attributes from one domain to another, e.g. [HFK+15, LTL10], this is usually modeled as a one-to-one association, that is, an attribute to an attribute. In our model, the association is more general in the sense that a category can be defined as a combination of attributes and related to a category in a different domain. One could imagine a more generic delegation graph that models the delegation of multiple categories, that is an association of a tuple of categories with another tuple of categories. We can also consider logical operators for the combination of categories in the tuple and thus create a language expressive enough to specify a wide range of delegations. However, even if theoretically possible, this seems quite difficult to be used in practice due to the complexity of finding an inter-domain agreement.

We do not take into consideration here the process of reaching this agreement. We assume that the delegation graph is statically defined beforehand according to a consensus between the different participants. It will be consulted by the delegation module to decide if the access request is accepted, after the access control application will have detected that a subject is invoking a service from another organization.

We model the delegation graph as a DAG since a user could possibly take advantage of cycles to get more privileges than he/she is allowed to. Moreover, unintended cycles could change the semantics of system authorization. When a cycle issue exists, we can treat it either at design-time, i.e. directly in the delegation graph, or at run-time, when the request is executed. Notice that, even in the presence of a cycle at design-time, this does not mean that the cycle is necessarily present at run-time, this depends on the access request that is executed. If there is a run-time cycle, we can specify a delegation tree for each execution path, so that the cycle is split into several paths and thus broken. Moreover, we can eliminate a cycle by defining more finely the domains and consider them as clusters of services. In this case, the nodes of the graph would represent services while the arcs represent category delegations from a service in a domain to a service in another domain.

In addition, in the real world, companies or organizations are often hierarchical (eg military hierarchy, seniority at work, etc.). We may assume that there is a hierarchy between domains and authorize the delegation only from the top of the hierarchy downwards, that is a domain can only delegate attributes to the domains below itself. If a (total) hierarchy is defined, then no loops are possible in the delegation chain.

The formal representation of the delegation graph is a logical predicate that takes four parameters *Delegate(org1, cat1, org2, cat2)* that means the category *cat2* from the organization *org2* delegates the category *cat1* to the organization *org1*. The delegation graph is of crucial importance to determinate access request decisions in the case of transitive service invocation. The derivation of permissions in the case of a service invoking another service outside of its organization is modeled by the following rule:

$$\forall org1, \forall org2, \forall cat1, \forall cat2, \forall s, \forall o, \forall act,$$
$$Empower(org2, s, cat2) \wedge Permission(org1, cat1, act, o) \wedge Delegate(org1, cat1, org2, cat2)$$
$$\leftarrow Is\_permitted(s, act, o) \quad (4.3)$$

The rule above means that a subject s has a permission to perform an action *act* on an object o if (i) s is associated to a category $cat2$ in the organization $org2$, (ii) a category *cat2* from the organization *org2* is mapped to the category *cat1* in the organization *org1* and (iii) the organization $org1$ grants the category $cat1$ the permission to perform the action *act* on the object o.

Following the original OrBAC specification, we use Datalog as the specification language in which to express our model. For evaluating an access request, firstly the system determines the subject's organization and evaluates his attributes in order to resolve a category for the subject. Then, it checks the organization who owns the requested resource. If the subject's organization is different from the resource organization, a delegation step is needed in order to determine her/his permissions. If transitive dependencies are present, the delegation chain is computed. The query returns *True* if access rights are correctly propagated through the

chain. It returns *False* if a denial of access is found somewhere in the path of involved services. Details on the pyDatalog [Car14] implementation and examples of access queries can be found in [Utt16, UBR14].

Properties of the model As a consequence of mapping authorization queries to Datalog queries, we can study the problem of answering authorization queries by reusing well-known results about Datalog (see [CGT89] for an overview on Datalog and [UBR15] for details on properties of our model). The main advantages of our analysis technique are two. First, there is no need to implement a prototype to experiment with the system behavior since it is possible to perform the simulation of the system from an abstract design before its deployment; thereby avoiding the difficulties of testing distributed applications and cutting down the costs of correcting errors on the implementation when the system is already in operation. Second, sometimes the code of certain integrated services may not be easily accessible or may cost money to be invoked; it is indeed desirable to be able to build an abstract specification that can be used to predict (some of) the outcomes of the system without incurring in extra costs for testing. Additionally, the environment in which the various services will be run may not be easily accessible; thereby making the testing of the composed system very difficult, if possible at all.

Proposition 4.1 *In our model,*

- *authorization queries admit only finitely many answers;*
- *answering authorization queries always terminates in polynomial time.*

Further security properties, like the consistency of an access policy modeled through our framework, can be verified by exploiting a rewrite specification (following the ideas of chapter3). In [Utt16] automated analysis is performed by using well-known rewriting tools such as AproVe [GSKT06] and CiME [CPU$^+$10] on two case studies, the research center detailed here as well as a medical center scenario.

Policy enforcement: extension of XACML architecture with delegation module In order to support the enforcement of the access control policies we have defined, we propose an extension of the XACML architecture. The extension consists in incorporating the construction of the delegation graph in the policy information point of the XACML architecture for keeping track of delegated authorization rights in transitive chains of service invocations. We have implemented the architecture on top of the WSO2[3].

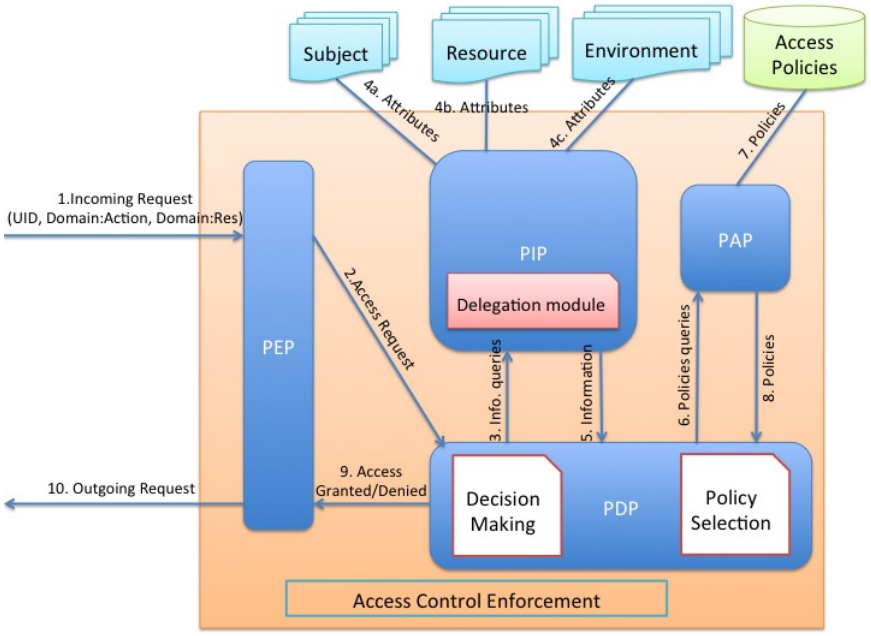


Figure 4.5: XACML Architecture with Delegation Graph Handling

In the following, we briefly discuss the interactions among the various XACML modules, as reported in Figure 4.5. The user's access request is intercepted by the Policy Enforcement Point (PEP), Step 1. The request and user's attributes are forwarded to the Policy Decision Point (PDP), Step 2, which asks the Policy Information Point (PIP) for additional informations about the subject. If the PDP determines from the policy and request that the user's organization is different from the object's organization, then delegation of category is needed. Then, the PIP will ask the delegation module, which contains the delegation gaph and the history of previous delegations, to compute a category for the user in the new organization. The delegation module first check the history of delegations of the user u making the request. If the category has been previously delegated to u in this organization, the module returns it to the PDP. Otherwise, it

[3] `http://wso2.com` an open source middle-ware platform for developing web service solutions.

will check the mapping of the user's category in the resource's organization from the delegation graph and then return it to PDP (Step 3-5). The PDP loads the XACML policy of the organization owning the resource from the Policy Administration Point (PAP) corresponding to the user's access request (Step 6-8). The PDP makes a decision and returns it to the PEP (Step 9). If the answer is positive, the PEP forwards the request to the transitively invoked service, otherwise the PEP throws an access denied exception (Step 10).

Using the presented XACML architecture, we have implemented and tested the case study described in the previous section as well as a case study on a clinical service management scenario [UBR15, UBR14].

4.1.3 Related Work

Nowadays, most web services in a SOA implementation use Identification-based access control, often with Federated Identity Management. This seems a natural solution, however there are several problems such as manageability, system evolution, transitivity and responsibility tracking (see [LK07] for a discussion on this). A reason for these difficulties is that FIdM does not address the federation of access policies which is much harder than federating identities.

In our work, we tackle the transitivity problem and we propose to solve it using an extension of a well-known standard as XACML. The transitive access problem occurs frequently in Web Service based collaborative systems since each organization provides various services that are protected by its own security policy. In [KL10] the authors propose a solution for transitive access problem by modeling the problem into authoriZation Based Access Control (ZBAC). Instead of the standard FIdM architecture, where the user identifies using his/her credentials via an Identity Management System in order to have access to a service, in ZBAC the user interacts directly with a policy engine which returns the set of authorization concerning the user upon receiving the his/her identity. Then, the user makes the request of a service including in the demand the rights he/she wants to exercise. Even if this approach allows one to avoid the problems associated with distributed identity management, it cannot be deployed using the most commonly adopted FIdM architecture. Moreover, it shifts at least part of the burden of

dealing with (a long list of) authorizations onto the user, who has to present the appropriate authorization to the server when making a request.

The work in [EBA+07] presents an access control model for SOA and its enforcement via an authorization verification service which is part of the IAM architecture. Access in the context of service composition, technically implemented using a BPEL engine, is briefly discussed and modeled as the sum of the access restrictions of the invoked web service operations. The solution we provide is at a more abstract level and is independent from the process execution environment. We develop a standard XACML architecture with an extended PIP, which can be easily integrated in a classical FIdM framework.

In [CON06] multiple policy domains and dynamic delegation of authority are considered. However, the authors do not specifically consider the problem of access request evaluation and access decision making in the case of transitive calls. The work in [SIM+07] addresses the problem of access control for web service composition. The access policies are specified in Pure-Past Linear Temporal Logic (PPLTL) that allows to exploit the history of service invocations to make access control decisions. Unfortunately in practice the specification of policies in PPLTL is not very friendly for security designers. Their access control model is implemented using a supply chain management (SCM) application.

[SYTB13] and [MOPB06] also discuss access control in web service composition. Nevertheless, their approach is different from ours. They consider the issue of service unavailability along a pathway to a target service, and they solve it by invoking dynamically alternative services belonging to different domains.

Several extensions of the OrBAC model have been proposed recently in order to specify security rules for intra- as well as inter-organizations. For instance [YD09] have proposed a new access control framework for inter-Organizational Web services (PolyOrBAC). The authors model permissions, prohibitions and obligations in timed automata to verify properties such as reachability and correctness. They consider the case of a service requested remotely from a different organization. In this case, in order to authorize a user from a different domain to access a service, a particular role is associated to a virtual user, then a specific rule defined

as a circumstance (context relation in OrBAC) is applied. However, they do not address the transitive access problem for dependent services, nor use requester's credentials for computing the rights of access.

4.2 Data Fusion processes

Cooperative systems often require large amounts of information gathered from a variety of sources, possibly controlled by different authorities, to properly function [CF18, TZG+13]. This information is typically processed and fused to create "new" data objects that can be further processed and shared to provide new services.

To ensure a high level of both security and business continuity in cooperative systems, fine-grained access control at the data level is needed. To implement such a level of granularity, one needs to keep track of *(i)* the evolution of data in the cooperative system, *(ii)* the security requirements and policies on these data, and *(iii)* past accesses to the data.

Unfortunately, traditional access control models are not expressive enough to cope with the security needs of multi-source cooperative systems. To address this gap, a range of novel access control models have been proposed in the last years. On the one hand, we can find history-based access control models [BDF05, EAC98, KNS08, SS04, TFS18] that account for the execution history (i.e., past accesses) for access decision making. These models have been further extended by exploiting provenance information about data objects and, thus, also accounting for their evolution [CKKT11, LB13, NPS13, SPNS16]. On the other hand, we can find policy frameworks for data fusion [dHZ16b, ZJW06] that aim to determine which access restrictions should be imposed on data objects based on the restrictions on the data used for their creation. However, to date there is no access control model that allows for the specification and enforcement of access constraints accounting for both the evolution of data objects and access restrictions on the underlying data fusion processes. Moreover, the aforementioned works usually provide ad-hoc policy languages to specify provenance-based access constraints. Therefore, these models are not supported by existing implementations, thus hindering their adoption on a large scale.

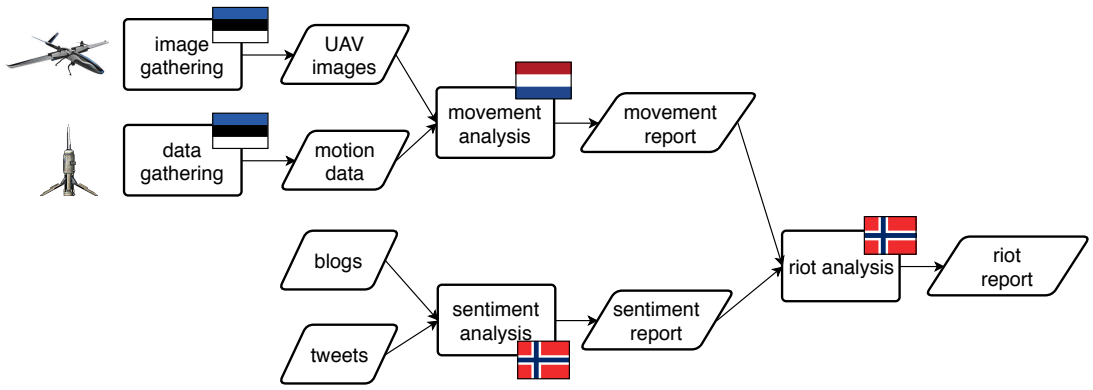


Figure 4.6: Data flow representing the generation of riot reports

4.2.1 Synthesis of our approach

In this work, we address the complex problem of access control for secure data fusion in cooperative systems. In particular, we present a unified access control framework that relies on data provenance to account for both the evolution and fusion of data objects.

We illustrates the challenges of regulating access to sensitive resources in cooperative systems through a running example within the military domain inspired from [dHZ16b].

Our scenario is set in Petros, an unstable country controlled by a military dictatorship. In recent years, protests against the dictatorial regime governing Petros have considerably grown. In response to protests, the government has mobilized the army, leading to an escalation of tension. This situation has attracted the attention of the international community and the NATO has created a Joint Task Force (JTF), named ALPHA, with the mission to restore peace in the territory. The JTF is a multinational operation composed of combat and intelligence units belonging to NATO countries, including the Neverland (NL), Eiseight (EE) and Noway (NO). As an illustrative scenario, we focus on a "hearts and minds" campaign within the JTF. A Neverland unit of the JTF is appointed to gather intelligence in a village; before approaching the village, the team leader of the unit needs information about possible riots in the village.

Figure 4.6 illustrates the process to gather intelligence in order to produce a riot report. The leader of the Neverland unit contacts an intelligence officer of the Noway army to get intelligence on the village. The intelligence officer uses an analysis tool to assess the threat of riots in the mission area using information from different sources. Riots are often characterized by crowds

of people in a certain area. To this end, the intelligence officer requests a movement report to the Command and Control Centre of the Neverland army. To assess the movement in the area, the latter requests UAV images and motion data collected by UAV and motion sensors deployed in the area by the Eiseight army. The intelligence officer also requests a sentiment report from an OSINT analyst of the Noway army. The OSINT analyst runs crawlers on open source blogs and tweets concerning the mission area and its inhabitants. The OSINT analyst provides the intelligence officer the sentiment report. The intelligence officer processes the collected information and fuses them in a riot report, which is sent to the team leader.

This scenario shows that international cooperation and orchestration is needed to ensure the success of the mission. In particular, information from heterogeneous sources should be collected and fused to enable situation awareness. Information sources, however, can be under the control of different authorities, and each authority might impose constraints on how and by whom its data can be accessed and processed. For example:

1. The Noway army might require a riot report to be accessed only by users with rank officer that participate to a certain mission and belong to the army of a country that has contributed to the creation of the requested report.

2. The Noway army might also require a riot report to be accessed only by users that are authorized to access all artifacts used for its creation.

3. The Noway army might impose that a riot report is created by a different user from the one who created the sentiment report used as input for the riot report.

Traditional access control models are not expressive enough to capture the access constraints above. We need policies that have "knowledge" about the process used for the creation of the intermediate and final artifacts. To specify those access constraints, we have identified the following criteria that a multi-source cooperation model should encompass:

C1: Multi-Perspective. Access constraints should not only account for the characteristics of the user requesting access (e.g., identity, role, rank, mission) but also for the characteris-

tics of the object to be accessed (e.g., type, security level), for the action to be performed and for the relations between these elements (policy 1).

C2: Artifact Evolution. Access constraints should account for the creation and evolution of objects within the system to determine the access permissions to be granted (policy 1). This includes the ability to constrain access permissions based on users' past actions as well as dynamic Separation and Binding of Duty (SoD/BoD) constraints (policy 3).

C3: Input Objects' Policies. Access constraints should account for the policies associated to the object(s) used for the generation of the requested object (policy 2).

In the next section, we present an access control model able to address the criteria above, based on the work published in [BBU17, BdHZ19].

4.2.2 Proposed model

To model and reason over provenance information, we adopt the Open Provenance model [MCF+11], which provides a core representation of provenance. Provenance information is usually represented as a labeled direct graph (called *provenance graph*) that expresses how a certain object was derived. A node of a provenance graph can be: an *artifact*, which is used to denote an immutable piece of state that can have a physical or digital representation; a *process*, which is used to denote the actions performed on an artifact and resulting in a new artifact; or an *agent*, which is used to denote the entity controlling or affecting the execution of a process. Hereafter, we use $\mathcal{N}$, $\mathcal{D}$, $\mathcal{P}$ and $\mathcal{S}$ to denote the sets of nodes, artifacts, processes and agents, respectively (with $\mathcal{N} = \mathcal{D} \cup \mathcal{P} \cup \mathcal{S}$ and $\mathcal{D}, \mathcal{P}, \mathcal{S}$ pairwise disjoint). Moreover, we use $\mathcal{R}$ to denote the set of role labels, which define the function of an agent or an artifact in a process.

The edges of the provenance graphs capture three types of role labeled casual dependencies (top of Table 4.2): $\mathsf{used} \subseteq \mathcal{P} \times \mathcal{D} \times \mathcal{R}$ captures, for processes, the artifacts they use; $\mathsf{wasGeneratedBy} \subseteq \mathcal{D} \times \mathcal{P} \times \mathcal{R}$ captures which artifacts are generated by which processes; $\mathsf{wasControlledBy} \subseteq \mathcal{P} \times \mathcal{S} \times \mathcal{R}$ captures, for processes, which agents control them. To consider a specific role or

ignore the role label of a casual dependency T, we respectively use $T^r(n, n') \triangleq T(n, n', r)$, where r is a role label, and $T^\circ(n, n') \triangleq \exists r \in \mathcal{R} : T(n, n', r)$ (for all $n, n' \in \mathcal{N}$).

The Open Provenance model includes several additional dependencies that can be derived from a provenance graph by composing dependencies (we use ";" to denote composition of casual dependencies). $\mathsf{wasTriggeredBy} \triangleq \mathsf{used}^\circ; \mathsf{wasGeneratedBy}^\circ$ captures that the execution of a process was triggered by another process and $\mathsf{wasDerivedFrom}^\circ \triangleq \mathsf{wasGeneratedBy}^\circ; \mathsf{used}^\circ$ captures that an artifact was derived from another artifact. Transitive closure $\mathsf{wasDerivedFrom}^+$ captures, for an artifact, all artifacts used to derive it, possibly indirectly. Similarly, $\mathsf{wasTriggeredBy}^+$ captures, for a process, all its direct and indirect triggering processes. The later is also helpful to capture indirect use and generation of artifacts: $\mathsf{used}^+ \triangleq \mathsf{wasTriggeredBy}^+; \mathsf{used}^\circ$ captures, for a process, all artifacts it indirectly depends on, whereas $\mathsf{wasGeneratedBy}^+ \triangleq \mathsf{wasGeneratedBy}^\circ; \mathsf{wasTriggeredBy}^+$ captures, for an artifact, all processes leading up to its generation.

Aiming to use provenance information to specify access requirements, we extend the Open Provenance model. For an artifact, we want to be able to refer to the role another artifact played in its creation. We thus label the Open Provenance model relation $\mathsf{wasDerivedFrom}^\circ$ by making the role explicit: $\mathsf{wasDerivedFrom}^r \triangleq \mathsf{wasGeneratedBy}^\circ; \mathsf{used}^r$ (for all $r \in \mathcal{R}$). We also introduce two custom dependencies that use a specific role $\mathsf{owner} \in \mathcal{R}$ to capture artifact owners and contributors. Following the classic assumption in discretionary access control where subjects own the objects they create [HRU76b], we say an owner of a process that generates an artifact owns that artifact: $\mathsf{owns} \triangleq \mathsf{wasGeneratedBy}^\circ; \mathsf{wasControlledBy}^{\mathsf{owner}}$. Actors owning any process indirectly involved in the creation of an artifact are considered contributors: $\mathsf{contributedTo} \triangleq \mathsf{wasGeneratedBy}^+; \mathsf{wasControlledBy}^{\mathsf{owner}}$. These relations are formally defined in Table 4.2.

Following the graphical notation defined in [MCF+11, MM13], we represent artifacts using circles, processes using rectangles and agents using octagons. Edges are annotated with the type of dependency and the role of artifacts and agents in processes. It is worth noting that the nodes in a provenance graph represent instances of agents, artifacts and processes. For instance, Figure 4.7 represents that: motion data *md*24 have been gathered from *motion sensor 1* and

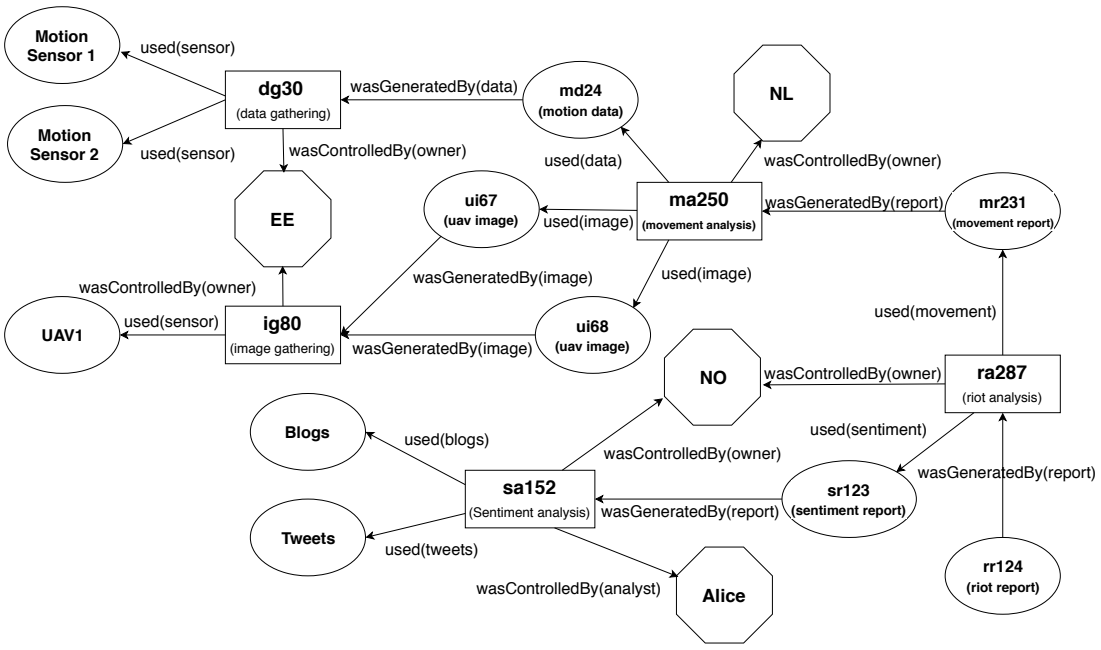


Figure 4.7: Provenance graph corresponding to the motivating example

Basic casual dependencies
$\mathsf{used}(p:\mathcal{P}, d:\mathcal{D}, r:\mathcal{R})$
$\mathsf{wasGeneratedBy}(d:\mathcal{D}, p:\mathcal{P}, r:\mathcal{R})$
$\mathsf{wasControlledBy}(p:\mathcal{P}, s:\mathcal{S}, r:\mathcal{R})$
$\mathsf{wasTriggeredBy}(p_1:\mathcal{P}, p_2:\mathcal{P}) \triangleq \exists d \in \mathcal{D}, r_1, r_2 \in \mathcal{R} : \mathsf{used}(p_1, d, r_1) \wedge \mathsf{wasGeneratedBy}(d, p_2, r_2)$
$\mathsf{wasDerivedFrom}^{\circ}(d_1:\mathcal{D}, d_2:\mathcal{D}) \triangleq \exists p \in \mathcal{P}, r_1, r_2 \in \mathcal{R} : \mathsf{wasGeneratedBy}(d_1, p, r_1) \wedge \mathsf{used}(p, d_2, r_2)$
Multi-step casual dependencies
$\mathsf{used}^{+}(p:\mathcal{P}, d:\mathcal{D}) \triangleq \exists p_i \in \mathcal{P}, r \in \mathcal{R} : \mathsf{wasTriggeredBy}^{+}(p, p_i) \wedge \mathsf{used}(p_i, d, r)$
$\mathsf{wasGeneratedBy}^{+}(d:\mathcal{D}, p:\mathcal{P}) \triangleq \exists p_i \in \mathcal{P}, r \in \mathcal{R} : \mathsf{wasGeneratedBy}(d, p_i, r) \wedge \mathsf{wasTriggeredBy}^{+}(p_i, p)$
$\mathsf{wasTriggeredBy}^{+}(p_1:\mathcal{P}, p_2:\mathcal{P}) \triangleq \mathsf{wasTriggeredBy}(p_1, p_2) \vee \exists p_i \in \mathcal{P} : \mathsf{wasTriggeredBy}(p_1, p_i) \wedge \mathsf{wasTriggeredBy}^{+}(p_i, p_2)$
$\mathsf{wasDerivedFrom}^{+}(d_1:\mathcal{D}, d_2:\mathcal{D}) \triangleq \mathsf{wasDerivedFrom}(d_1, d_2) \vee \exists d_i \in \mathcal{D} : \mathsf{wasDerivedFrom}(d_1, d_i) \wedge \mathsf{wasDerivedFrom}^{+}(d_i, d_2)$
Custom casual dependencies (not in the Open Provenance model)
$\mathsf{wasDerivedFrom}(d_1:\mathcal{D}, d_2:\mathcal{D}, r:\mathcal{R}) \triangleq \exists p \in \mathcal{P}, r_i \in \mathcal{R} : \mathsf{wasGeneratedBy}(d_1, p, r_i) \wedge \mathsf{used}(p, d_2, r)$
$\mathsf{owns}(s:\mathcal{S}, d:\mathcal{D}) \triangleq \exists p \in \mathcal{P}, r \in \mathcal{R} : \mathsf{wasGeneratedBy}(d, p, r) \wedge \mathsf{wasControlledBy}(p, s, \text{`}owner\text{'})$
$\mathsf{contributedTo}(s:\mathcal{S}, d:\mathcal{D}) \triangleq \exists p \in \mathcal{P} : \mathsf{wasGeneratedBy}^{+}(d, p) \wedge \mathsf{wasControlledBy}(p, s, \text{`}owner\text{'})$

Table 4.2: Casual dependencies in the provenance graph

motion sensor 2 through the instance of the data gathering process $dg30$; riot report $rr124$ has been generated from movement report $mr231$ and sentiment report $sr123$ through the instance of process riot analysis $ra287$; and so on. For the sake of clarity, in the figure the type of artifacts and processes is reported in parenthesis.

Policy representation For representing our scenario access requirements, we rely on the attribute-based access control (ABAC) paradigm, that we extend with provenance-based access constraints. In ABAC, policies and access requests are defined in terms of attribute name-value

pairs. A query $q = \{(a_1, v_1), \ldots, (a_k, v_k)\}$ is a set of attribute name-value pairs. Formally, let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be a finite set of attributes. Given an attribute $a \in \mathcal{A}$, $\mathcal{V}_a$ denotes the domain of a. The set of queries $\mathcal{Q}_\mathcal{A}$ is defined as $\wp(\bigcup_{i=1}^{n} a_i \times \mathcal{V}_{a_i})$ and a query $q = \{(a_1, v_1), \ldots, (a_k, v_k)\}$ is a set of attribute name-value pairs (a_i, v_i) such that $a_i \in \mathcal{A}$ and $v_i \in \mathcal{V}_{a_i}$. We also assume an infinite set of variables. Each variable x has a domain $\mathcal{V}_x \subseteq \mathcal{N}$.

Example 1 *The access request made by the team leader of the Neverland unit can be modeled as query*

$$\begin{aligned} q_{rr} = \{&(\mathsf{subject\_id}, \mathit{Paul}), (\mathsf{subject\_rank}, \mathit{officer}), (\mathsf{subject\_army}, \mathit{NL}),\\ &(\mathsf{subject\_mission}, \mathit{alpha}), (\mathsf{action}, \mathit{read}), (\mathsf{resource\_id}, \mathit{rr124}),\\ &(\mathsf{resource\_type}, \mathit{riot\_report})\} \end{aligned}$$

*This query states that Paul, an officer of the Neverland army participating to mission ALPHA, wants to read the riot report identified by code rr*124*.*

For policy specification, we adopt a language $\mathcal{P}ol_\mathcal{A}$ that provides a compact representation of XACML [OAS13] with the addition of path conditions. For details the reader can refer to [BdHZ19]. A path condition can be a casual dependency in the provenance graph (as the ones defined in Table 4.2), or can be constructed from casual dependencies using logical operators (e.g., $\neg$, $\wedge$, $\vee$). As in XACML, a policy can be a decision, either *permit* (1) or deny (0), a target policy (t, pol) where the target t defines the applicability of the policy, or a composite policy $ca(pol_1, \ldots, pol_n)$ with ca a combining algorithm. Here, we consider standard combining algorithms [OAS13]: *permit-overrides* (pov), *deny-overrides* (dov) and *first-applicable* (fa). We also define policy references, that are used to retrieve policies associated with objects involved in the query. A policy reference *pol_ref* is an expression that resolves into a *uri* pointing at a policy. We do not detail here how to specify and resolve policy references but we simply assume that a policy reference *pol_ref* yields a function $uri : \mathcal{N}^k \to \mathcal{P}ol_\mathcal{A}$.

Example 2 *Consider a policy involving the entities in the provenance graph of Figure 4.7 that*

includes policy reference[4]

$$\mathsf{dov}(\{uri(x,y) \mid \mathsf{wasDerivedFrom}^{+}(\mathsf{resource\_id}, x) \wedge \mathsf{owns}(y,x)\})$$

This policy reference can be resolved with a function

$$uri(x,y) = \mathsf{uri{:}policies.alpha.jtf.nato.org?resource\_id} = x \,\&\mathsf{agent} = y$$

which encodes a Uniform Resource Identifier (URI) that identifies the policies applicable to the nodes in the provenance graph that satisfy the given query. The string after the question mark (?) represents the query component of the URI.

Given an applicable request $\{(\mathsf{resource\_id}, mr231), \ldots\}$ *for movement report mr231, the policy reference resolves to*

$$\mathsf{dov}(\{uri(md24, EE), uri(ui67, EE), uri(ui68, EE)\})$$

with $uri(md24, EE)$ *returning the EE policy applicable to motion data, and* $uri(ui67, EE)$, $uri(ui68, EE)$ *returning the EE policy applicable to its UAV gathered images.*

For a request $\{(\mathsf{resource\_id}, rr124), \ldots\}$, *it would also consider the NL policy that applies to mr231 and the NO policy that applies to sr231.*

The policy language $\mathcal{P}ol_{\mathcal{A}}$ allows the definition of three main types of access constraints:

1. *attribute-based constraints* that define permissions and prohibitions based on subject, action and object attributes, thus accounting for multi-perspectives in access decision making (**C1**);

2. *provenance-based constraints* that define permissions and prohibitions based on path conditions over the provenance graph, thus accounting for the evolution of artifacts (**C2**);

[4]Note that $uri(x_1, \ldots, x_n)$ is considered to bind variables $x_1, \ldots, x_n$ in *path*.

3. *input-dependent constraints* that define permissions and prohibitions based on the policies of the artifacts and processes used (possibly indirectly) in the generation of new artifacts (**C3**). These constraints are modeled using policy references pointing to (external) access control policies identified via a path condition in the provenance graph.

In the following example, we demonstrate how policy language $\mathcal{P}ol_{\mathcal{A}}$ can be used to model the access control policies underlying our military scenario.

Example 3 *The access control policy* p_{rr} *used in our motivating example to regulate the access to objects of type* riot report *(policies 1 & 2) can be formalized as follows:*

$$
\begin{aligned}
pol_{rr} &= (\mathsf{resource\_type} = riot\_report, \mathsf{dov}(pol_A, pol_B)) \\
pol_A &= \mathsf{pov}(pol_1, pol_2, 0) \\
pol_B &= \mathsf{pov}(pol_3, 0) \\
pol_1 &= (\mathsf{subject\_rank} = officer \wedge \\
&\qquad \mathsf{subject\_mission} = alpha \wedge \mathsf{action} = read, 1) \\
pol_2 &= (\mathsf{contributedTo}(\mathsf{subject\_army}, \mathsf{resource\_id}) \wedge \\
&\qquad (\mathsf{action} = read \vee \mathsf{action} = download), 1) \\
pol_3 &= \mathsf{dov}(\{uri(x) \mid \mathsf{wasDerivedFrom}^{\circ}(\mathsf{resource\_id}, x)\}
\end{aligned}
$$

Policy p_{rr} *combines two composite policies using the* deny-overrides *combining algorithm, one* (pol_A) *specifying access constraints specific to riot reports and one* (pol_B) *used to refer to the policies of the object(s) used for its creation. In particular,* pol_A *comprises two policies,* pol_1 *and* pol_2*, and a default* Deny *policy (represented by* 0*). The first policy* pol_1 *encodes an attribute-based constraint stating that a subject having rank* officer *in her/his national army and participating to mission* alpha *is allowed to read the riot report. Policy* pol_2 *encodes a provenance-based constraint stating that access to the report is allowed if the subject belongs to (the army of) a country that has contributed to the creation of the riot report. The notion of* contributedTo *is formalized as a path condition over the provenance graph, meaning that the country has participated to at least one activity in the whole process leading to the creation of the riot report.*

Policy pol_B comprises policy pol_3 (and a Deny *default policy). In particular, pol_3 specifies an input-dependent constraint in the form of a reference to the (external) policies pol_x associated with the artifacts used in the creation of the riot report, in our case movement report $mr231$ and sentiment report $sr123$ (Figure 4.7). This reference is defined by a path condition (*wasDerivedFrom*) on the provenance graph taking as arguments the id of the resource from the access query, i.e. $rr124$, and variable x. The policies associated with the movement and sentiment reports are combined with the* deny-overrides *combining algorithm. Accordingly, the subject is allowed to access the riot report if he can access all artifacts used in the creation of the report.*

Policy evaluation Given a set of policies $\mathcal{P}ol_{\mathcal{A}}$, a set of queries $\mathcal{Q}_{\mathcal{A}}$ and a set of decisions $\mathcal{D}$, a policy evaluation function is a function $[\![\cdot]\!]_P : \mathcal{P}ol_{\mathcal{A}} \times \mathcal{Q}_{\mathcal{A}} \to \mathcal{D}$ such that, given a query q and a policy pol, its semantics $[\![pol]\!]_P(q)$ represents the decision of evaluating pol against q. For the sake of simplicity, here we assume $\mathcal{D} = \{Permit, Deny, NA\}$ (where NA stands for Not Applicable) but the provided semantics can be easily extended to deal with the *Indeterminate* decision supported by XACML [OAS13].

The formal semantics for policy evaluation is defined in [BdHZ19] and follows standard XACML semantics. We report here only the interesting cases, related in particular to path evaluations.

We evaluate node expressions to sets of nodes, using the evaluation function $[\![\cdot]\!]_N$. When a variable is used as a node it should always be bound (e.g. in a path expression). If a free variable x occurs, we assume that it does not match any node, i.e. $[\![n]\!]_N(q) = \{n\}$ and $[\![x]\!]_N(q) = \emptyset$. Node expressions could match multiple nodes, as an attribute a may take multiple values in q. Specifically, attributes are instantiated to the nodes corresponding to the attribute values in the query: $[\![a]\!]_N(q) = \{n \in V_a \mid (a, n) \in q\}$. Targets evaluate to Booleans using the evaluation function $[\![\cdot]\!]_T$. Similarly to nodes expressions, for evaluating (a, v), we look for any matching value of a in the request q: $[\![op(a, v)]\!]_T(q)$ is *True* if $\exists v' \in V_a : (a, v') \in q \wedge \boldsymbol{op}(v', v)$, or *False* otherwise.

Path conditions evaluate to Booleans using the evaluation function $[\![\cdot]\!]_C$. In particular, we eval-

uate casual dependencies $\mathbf{T}^{\circ}$ and $\mathbf{T}^{r}$ considering any possible values for the node expressions:

$$[\![T^{\circ}(node, node')]\!]_C(q) = \begin{cases} \mathit{True} & \text{if } \exists n \in [\![node]\!]_N(q), \\ & n' \in [\![node']\!]_N(q) : \mathbf{T}^{\circ}(n, n') \\ \mathit{False} & \text{otherwise} \end{cases}$$

$$[\![T^{r}(node, node')]\!]_C(q) = \begin{cases} \mathit{True} & \text{if } \exists n \in [\![node]\!]_N(q), \\ & n' \in [\![node']\!]_N(q) : \mathbf{T}^{r}(n, n') \\ \mathit{False} & \text{otherwise} \end{cases}$$

The logical operators (in paths and targets) have their usual interpretation.

Finally, policies evaluate to a decision in $\mathcal{D}$. Policy (t, pol) is not applicable if t does not evaluates to *True*; otherwise its decision is that of pol. Policy $ca(pol_1, ..., pol_m)$ is evaluated by executing the combining algorithm **ca** on the decisions $[\![pol_1]\!]_P(q)$ through $[\![pol_m]\!]_P(q)$. The semantics of combining algorithms is the same as in standard XACML. For policies containing a policy reference, we define $[\![\cdot]\!]_P$ as follows

$$[\![ca(\{pol\_ref(x_1, \ldots, x_k) \mid path\})]\!]_P(q) = \mathbf{ca}([\![pol_1]\!]_P(q), \ldots, [\![pol_m]\!]_P(q))$$

where $\{pol_1 \ldots, pol_m\} = \{uri(n_1, \ldots, n_k) \mid n_1 \in \mathcal{V}_{x_1}, \ldots, n_k \in \mathcal{V}_{x_k}, [\![path[n_1/x_1, \ldots, n_k/x_k]]\!]_C(q) = \mathit{True}\}$

Thus, to evaluate $ca(\{pol\_ref(x_1, ..., x_k) \mid path\})$, we find any nodes for $x_1, \ldots, x_k$ that make the path evaluate to *True*, get the corresponding policies using the policy reference function *uri* and combine the decisions for these policies with **ca**. Note that the order of the policies does not matter for the combining algorithms we allow. We give an example of policy evaluation below.

Example 4 *Consider the access control policy pol_{rr} defined in Example 3 and the access query q as defined in Example 1. First of all, we have $[\![(\mathsf{resource\_type} = riot\_report)]\!]_T(q) = True$, thus*

the policy pol_{rr} *is applicable for the query* q *and we have* $[\![p_{rr}]\!]_P(q) = \mathsf{dov}([\![pol_A]\!]_P(q), [\![pol_B]\!]_P(q))$. *For policy* pol_A *it is not difficult to see that we obtain* $[\![pol_A]\!]_P(q) = Permit$. *For policy* pol_B *we have* $[\![pol_B]\!]_P(q) = \mathsf{pov}([\![pol_3]\!]_P(q), [\![0]\!]_P(q)) = \mathsf{pov}(\mathsf{dov}([\![pol_{sr}]\!]_P(q), [\![pol_{mr}]\!]_P(q)), Deny)$ *with* pol_{sr} *and* pol_{mr} *the policies associated to the sentiment and the motion reports, respectively. Supposing* $[\![pol_{sr}]\!]_P(q) = Deny$ *as the Netherlands Army did not participate to the creation of the sentiment report, we have* $[\![pol_B]\!]_P(q) = \mathsf{pov}(Deny, Deny) = Deny$ *and thus finally* $[\![p_{rr}]\!]_P(q) = \mathsf{dov}(Permit, Deny) = Deny$.

Policy enforcement: extension of XACML architecture with provenance module The evaluation and enforcement of our framework can be realized within existing access control mechanisms. Specifically, we conservatively extend the XACML reference architecture as shown in Figure 4.8: in addition to the XACML architecture components, we introduce a *Provenance Module* that encompasses: *(i)* the provenance graph generated from the system logs, *(ii)* the definition of casual dependencies, i.e. capturing Table 4.2, and *(iii)* a provenance reasoner for evaluating path queries, which are path conditions in which attributes have been instantiated to the value they take in the request. The Provenance Module is invoked whenever in the access control policy there is a path condition to evaluate. If the evaluation of a policy requires evaluating a path condition, the Policy Decision Point (PDP) asks the Provenance Module to resolve the corresponding path query (corresponding to steps 9 and 10 in the picture). Recall that our framework supports two kinds of path conditions: *(a)* Boolean path conditions (e.g., $\mathsf{contributedTo}(\mathsf{subject\_army}, \mathsf{resource\_id})$ like in policy pol_2) and *(b)* policy reference path conditions (e.g., pol_3 of our example). In the first case, the Provenance Module takes the path query as input and returns *true* if there exists a path in the provenance graph that matches the path query; otherwise, it returns *false*. Path queries representing policy references are resolved through a two-step process: first, the module runs a path matching algorithm to return all entities that satisfy the path query in the provenance graph, and second the policies associated to these entities are retrieved and returned for evaluation. We have implemented the Provenance Module in Prolog [Llo84]. This allows us to benefit from the efficient reasoning capabilities provided by Prolog solvers for path condition resolution rather than relying on an

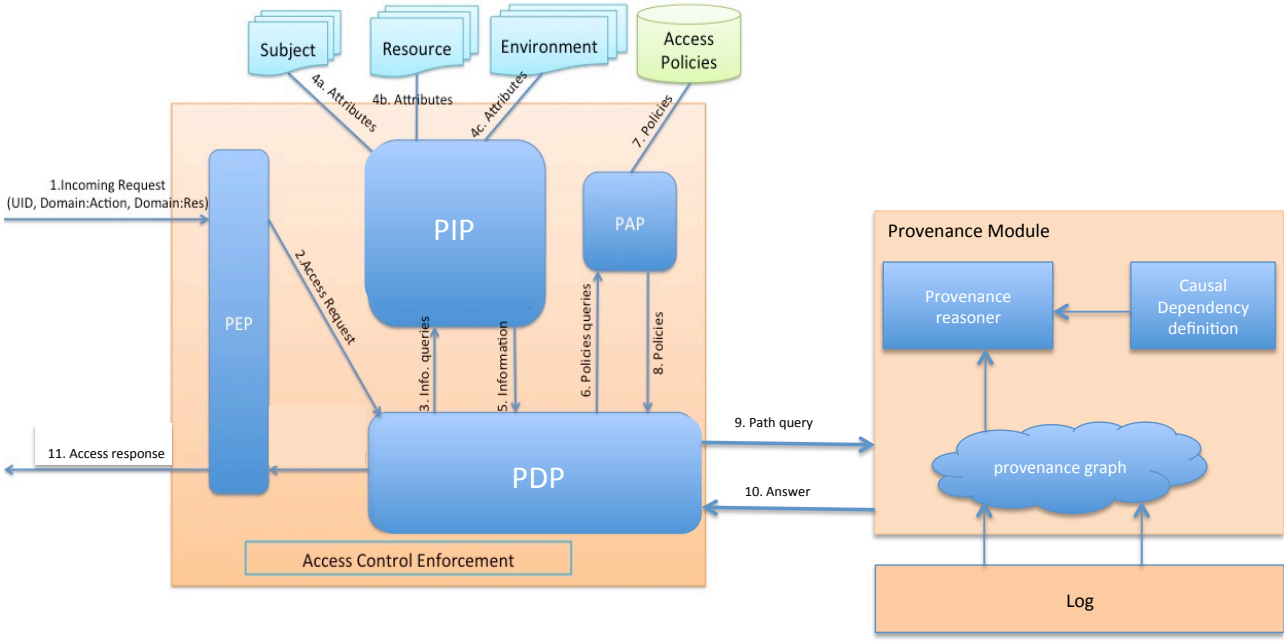


Figure 4.8: Architecture of the provenance-based access control mechanism

ad-hoc algorithm. In particular, we implemented rules both for constructing the provenance graph from a given access log and for resolving path queries based on the obtained provenance graph. Intuitively, the first set of rules are used to infer basic casual dependencies (i.e., used, wasGeneratedBy and wasControlledBy) from the access log, whereas the latter is used to derive the other standard and custom casual dependencies as defined in Table 4.2.

Path queries are verified by the Provenance Module through Prolog queries. Upon receiving a request to resolve a path query from the PDP, the Provenance Module constructs the corresponding Prolog query (i.e., the goal to be proved) and interacts with the underlying Prolog solver, which uses its knowledge of deduction rules (i.e., the program consisting of the rules for constructing casual dependencies) to prove or to refute the goal. When a path condition is used for policy referencing, the correspond query might contain unbounded variables. In this case, Prolog solvers enumerate all the values of these variables (which correspond to nodes in

the provenance graph) that satisfy the query. The answers to the query are used to fetch the relevant policies from the Policy Administration Point (PAP), which are evaluated by the PDP to determine whether access should be granted or denied.

4.2.3 Related Work

Our work is related to history-based access control, provenance-based access control and access control for data fusion. The past history of the system is at the core of authorization decision making in our model. This is also the main feature of history-based access control systems, which can be grouped into two main families: dynamic history-based access control systems in which programs are monitored at run-time [KNS08, TFS18] and static history-based access control systems in which one (statically) verifies that (an approximation of) the program's behavior will always conform to the policy (using, e.g., type systems [BDF05] or model checking [SS04]). Our approach is more closely related to the first type as it uses provenance information for decision making. Krukow et al. [KNS08] present a logical framework for behavior-based decision-making in which policies define access requirements on the past behavior of principals that must be fulfilled in order for an action to be authorized. The framework comprises a formal model of behavior, based on event structures, a declarative logical language for specifying properties of past behavior and an automata-based algorithm for checking whether a particular behavior satisfies a property expressed in the given language. The notion of security automata was firstly introduced in [Sch00], where the policy is given in terms of an automata and safety properties are enforced using a (automata-based) reference monitor that tracks execution history. Automata are also adopted in [TFS18] where access control policies prescribe constraints on the order in which permissions should be exercised. These works usually focus on the principals' past behavior and do not consider the evolution of the object to be accessed. On the contrary, our work also exploits information on how and by whom the object of interest has been generated for access decision making. In this respect, our approach is closer to provenance-based approaches, discussed next.

A number of works [CKKT11, LB13, NPS13, PY14, SPNS16] propose to exploit provenance

information in access control. Sun et al. [SPNS16] define an access control model that uses provenance information for access decision making and this is modeled by including provenance paths in the rules forming a policy. This work was later extended in [NPS13] where dynamic SoD constraints are supported by adding contextual attribute information in the provenance graph. A similar model is presented in [LB13] where provenance is combined with RBAC and applied to a cloud environment. Cadenhead et al. [CKKT11] extend an XML-based language for the specification of ABAC policies with regular expression and use SPARQL to query the provenance graph (represented as RDF triples). Similarly to these works, we use provenance information to express access constraints. However, our work differs from these approaches in several ways. First, our formalization of provenance-based access constraints can be encoded in XACML and their evaluation requires minimal changes to existing XACML implementations. Moreover, we also exploit provenance for referencing external policies in order to support data fusion (**C3**), which is a new feature comparing to previous provenance-based access control models. Pei and Ye [PY14] propose a policy retrieval framework based on provenance. This framework exhaustively generates candidate policies for each artifact role related to the target action type by enumerating all possible combinations of dependency type and policy combining algorithm. Then, each candidate policy is verified against provenance and recorded along with the corresponding access control decision. These training examples are fed into a decision tree classifier to derive the provenance-based access control policies to be enforced. Our framework differs from the one of Pei and Ye in the fact that it uses provenance to identify the policies applicable to the artifacts used in the creation of the object of interest.

To the best of our knowledge, only few works have proposed to account for the policies associated to the artifacts used in the generation of the requested object [dHZ16b, ZJW06]. In particular, den Hartog et al. [dHZ16b] propose a policy framework to regulate data fusion processes and the artifacts obtained from these processes. This framework makes use of policy templates to define how the policies of the artifacts used in data fusion processes should be combined. However, it only considers the direct inputs for an artifact, thus only providing limited support for **C2**. Moreover, these policies are static in the sense that they are defined when an object is created while the link with the input artifacts is not maintained. On the other hand, in

our work we exploit the provenance graph to maintain the link between an artifact and the artifacts used for its creation and use policy references to retrieve the policies of those artifacts at evaluation time.

Our framework relies on provenance graphs to determine user permissions. The use of graphs for access decision making can be found also in community-based systems and, in particular, online social networks. In these systems, access decisions are based on interpersonal relationships between users, which are modeled in a social graph. Several relationships-based access control (ReBAC) models have been proposed (see [PSZ18, Section 3] for a survey). These models define permissions in terms of paths in the social graph (e.g., friends, friends of friends) as well as enable the use of topology policies to specify access restrictions based on topological properties of the social graph (e.g., degree of separation, k-clique). However, these models only account for relationships between users and do not consider the relations between users and objects that is requested. This limitation is addressed by Crampton and Sellwood [CS14], who propose a generic ReBAC model based on path conditions. A path condition in [CS14] is modeled as a sequence of relationships, which is used to map the resource requester to a (set of) authorization principal(s). The subject and resource specified in the access request are first used to find the set of applicable principals. These principals are then used to determine whether the action specified in the request should be authorized based on a given authorization policy. This model, however, only represent a static view of the system and does not account for processes and, in general, for the creation and evolution of artifacts.

Chapter 5

Access monitoring in Business Processes

The notions of *business process* and *workflow* are tightly interwoven and often represent complex procedures. Following [WFM96] a business process is "a set of linked activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships", and a workflow is the "automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules". These definitions show that many different concerns contribute to what a business process is and how it is automated. Usually these aspects are projected along different viewpoints such as the control flow of a workflow, the resources necessary to its enactment, etc. An additional dimension is given by security-related dependencies which are specified as additional constraints on the execution of the various tasks. A workflow task is executed by a user who should be entitled to do so; e.g., the teller of a bank may create a loan request whereas only a manager may accept it. Besides, additional authorization constraints are usually imposed on task execution, such as Separation of Duty (SoD) or Binding of Duty (BoD) whereby two distinct users or the same user, respectively, must execute two tasks.

The workflow satisfiability problem (WSP for short), as introduced in Section 2.2.3 and detailed

in Section 5.2, is known to be NP-hard already in the presence of one SoD constraint [WL10].

Different solutions to the WSP consider at least two characteristics: if the order of the tasks is considered and if satisfiability is checked at design-time (before the execution of any instance of the workflow) or at run-time (during execution). The separation between ordered and unordered WSP was presented in [CG13b]. A classification of WSP approaches in the design-time/run-time dimension was done in a recent survey [HAM15]. Design-time techniques ensure the existence of at least one satisfying assignment, whereas run-time techniques enforce that a workflow instance follows a satisfying execution. It is possible in principle to use, at run-time, an algorithm that statically solves the WSP, but this is very inefficient, as it entails solving a new instance of the problem for each user request. Although there are current solutions to the problem, they are either not focused on its run-time version which we discuss in this chapter (see, e.g., [HAM15]) or not precise [Kar15].

5.1 Synthesis of our approach

The idea underlying our approach can be summarized as follows. To solve the WSP, we use the capability of model checkers to return counterexamples as follows. We formally represent security-sensitive workflows as symbolic transition systems. A symbolic model checker is then asked to find a counterexample to the property that the system is not terminating. Indeed, the returned counterexample (if any) is precisely an execution scenario solving the WSP. Since we are interested in finding all execution scenarios, we modify the model checker in order to compute all counterexamples, not just one. We represent a set of counterexample scenarios by using a reachability graph, i.e., a directed graph whose edges are labeled by task-user pairs in which users are symbolically represented by variables and whose nodes are labeled by a symbolic representation (namely, a formula of first-order logic) of the set of states from which it is possible to reach a state in which the workflow successfully terminates. The graph allows us to compactly encode all possible interleavings of tasks in a workflow. From the set of formulae labeling the nodes in the reachability graph we derive a monitor capable of answering

positively user requests to execute tasks at run-time iff the user is authorized to do so by the policy, there is no violation of authorization constraints and the workflow can still terminate (i.e., the request is part of one of the scenarios computed in the reachability graph). This graph is refined in [dSRCP15] to find execution scenarios that satisfy properties defined by the user.

The crux of our approach is that the model of the security-sensitive workflow in input to the symbolic model checker only contains the constraints on the control-flow and the authorization constraints, while it abstracts away from the authorization policy. In this way, the reachability graphs computed by the model checker can then be refined with respect to an authorization policy associated with a particular deployment context.

We also propose a variant of WSP by considering positive weights associated with transitions, if these are not compliant with the specified authorization policy or constraints. Indeed in certain situations, business continuity has to be preferred to security requirements, so a methodology to guarantee the termination of the workflow with "minimal" violations is needed by security administrators. Note that is not possible to reuse our solution for the WSP to solve the weighted version of WSP because the pre-computed reachability graph does not consider constraint violations.

This chapter describes the results published in [BdSR18, BR13b, BR13a, BdSR15]

5.2 Workflow Satisfiability Problem

The *Workflow Satisfiability Problem* (WSP) consists in checking if there exists an assignment of users to tasks such that a security-sensitive workflow successfully terminates while satisfying all authorization constraints.

To formalize the WSP, we need first to introduce some preliminary notions.

Scenarios and workflows Given a finite set T of tasks and a finite set U of users, an *execution scenario* (or, simply, a *scenario*) is a finite sequence of pairs of the form (t, u)—also

written as $t(u)$—for $t \in T$ and $u \in U$. The intuitive meaning of a scenario $\eta = t_1(u_1), \ldots, t_n(u_n)$ is that task t_i is executed before task t_j for $1 \leq i < j \leq n$ and that task t_k is executed by user u_k for $k = 1, \ldots, n$. Among the scenarios in a workflow, we are interested in those that describe successfully terminating executions. Since the notion of successful termination depends on the application, from now on we consider only successfully terminating behaviors scenarios. A *workflow* $W(T, U)$ is a (finite) set of scenarios.

Authorization relation Given a workflow $W(T, U)$, an *authorization relation* TA is a sub-set of $U \times T$ where $(u, t) \in TA$ means that u is entitled to execute task t. Following [CHK14], we call *authorized* a scenario η of a workflow $W(T, U)$ according to TA iff (u, t) is in TA for each $t(u)$ in η.

Authorization constraint An *authorization constraint* over a workflow $W(T, U)$ is a tuple $(t_1, t_2, \bowtie)$ where $t_1, t_2 \in T$ and $\bowtie$ is a sub-set of $U \times U$. For instance, a SoD constraint between tasks t and t' can be formalized as $(t, t', \neq)$ with $\neq$ being the complement of the identity relation over U. A scenario η of $W(T, U)$ *satisfies* the authorization constraint $(t_1, t_2, \bowtie)$ over $W(T, U)$ iff for $t_1(u_1)$ and $t_2(u_2)$ in η we have that $(u_1, u_2) \in \bowtie$. We call *eligible (according to a set K of authorization constraints)* a scenario η of a workflow $W(T, U)$ iff η satisfies K (i.e. satisfies each constraint of K).

Definition 5.1 (Security Sensitive Workflow) *Following [AP10], we call* security-sensitive workflow *(SSW) the triple $(W(T, U), TA, K)$ where $W(T, U)$ is a workflow, TA an authorization relation, and K a (finite) set of authorization constraints.*

Definition 5.2 (WSP Problem) *Given a SSW $(W(T, U), TA, K)$, the WSP consists of checking whether there exists an execution scenario in $W(T, U)$ which is both authorized and eligible.*

Such a problem has been studied in several papers; see, e.g., [WL10, PYL13]. The run-time version of the WSP consists of answering sequences of user requests at execution time and

ensuring successful termination together with the satisfaction of authorization constraints. This problem has received less attention and only an approximate solution is available [BBK12a].

5.3 Monitoring security-sensitive workflows

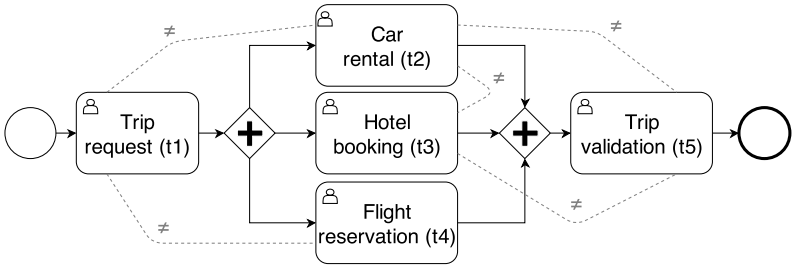


Figure 5.1: Workflow in extended BPM notation

We describe our approach to synthesize run-time monitors for security-sensitive workflows on a trip request process. The workflow is composed of five tasks—each one indicated by a box labeled by Trip request ($t1$), Car rental ($t2$), Hotel booking ($t3$), Flight reservation ($t4$), and Trip validation ($t5$)—whose execution is constrained as follows (cf. solid arrows and diamonds labeled with $+$): $t1$ must be executed first, then $t2$, $t3$ and $t4$ can be executed in any order, and when all have been performed, $t5$ can be executed, thereby terminating the workflow. Additionally, each task is executed under the responsibility of a user (indicated by the small icon inside the boxes corresponding to the various tasks) who has the right to execute it according to some access control policy—not shown in Figure 5.1—and the five authorization constraints depicted as dashed lines labeled by the symbol $\neq$ for Separation of Duty (SoD). So, for example, the authorization constraint connecting the boxes of $t1$ and $t2$ requires the user executing $t2$ to be distinct from the one that has executed $t1$, i.e. the user who requests the trip cannot also rent a car.

Our goal is to synthesize a run-time monitor, capable of ensuring that all execution and authorization constraints are satisfied. Our approach is organized in two phases: off-line and on-line.

Off-line. We first construct a symbolic transition system S whose executions correspond to

those of the security-sensitive workflow. Then, we use a symbolic model checker to explore all possible terminating executions of the workflow which satisfy both the causality and the authorization constraints. We assume the model checker to be able to return a symbolic representation R of the set of all states, called reachable, encountered during the exploration of the terminating executions of S. We use particular classes of formulae in first-order logic to be the symbolic representations of S and R.

On-line. We derive a Datalog program M from the formulae R, representing the set of states reachable in the terminating executions of S and the policy P specifying which user can perform which task. The Datalog program M derived in this way is the monitor capable of guaranteeing that any request of a user to execute a task is permitted by P, satisfies the authorization constraints (such as SoD), and the workflow can terminate its execution.

We illustrate the two phases on the security-sensitive workflow in Figure 5.1.

5.3.1 Off-line phase

First of all, we build the symbolic transition system S in two steps: (i) we adopt the standard approach (see, e.g., [vdAH03]) of using (extensions of) Petri nets [Mur89] to formalize the semantics of workflows and (ii) we adapt the well-known translation of Petri nets to symbolic transition systems (see, e.g., [SSM03]) to the class of extended Petri nets used in this paper.

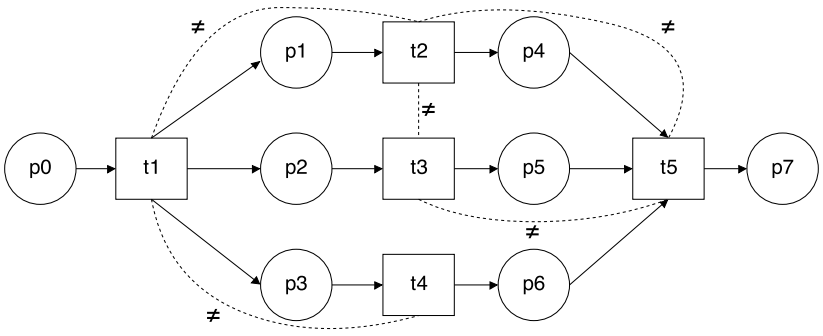


Figure 5.2: Workflow as an extended Petri net

Figure 5.2 shows the extended Petri net that can be automatically derived from the BPM notation of Figure 5.1. Tasks are modeled as transitions or events (the boxes in the figure)

whereas places (the circles in the figure) encode their enabling conditions. At the beginning, there will be just one token in place $p0$ which enables the execution of transition $t1$. This corresponds to the execution constraint that task $t1$ must be performed before all the others. The execution of $t1$ removes the token in $p0$ and puts a token in $p1$, another in $p2$, and yet another in $p3$; this enables the execution of $t2$, $t3$, and $t4$. Indeed, this corresponds to the causality constraint that $t2$, $t3$, and $t4$ can be executed in any order after $t1$ and before $t5$. In fact, the executions of $t2$, $t3$, and $t4$ remove the tokens in $p1$, $p2$, $p3$ and put a token in $p4$, $p5$, and $p6$ which, in turn, enables the execution of $t5$. This removes the token in $p4$, $p5$, $p6$ and put a token in $p7$ which enables no more transitions. This corresponds to the fact that $t5$ is the last task to be executed. The fact that there is at most one token per place is an invariant of the Petri net. This allows us to symbolically represent the net as follows: we introduce a Boolean variable per place (named as the places in Figure 5.2) together with a Boolean variable representing the fact that a task has already been executed (denoted by d_t and if assigned to true implies that task t has been executed). So, for instance, the enabling condition for the execution constraint on task $t1$ can be expressed as $p0 \wedge \neg d_{t1}$ meaning that the token is in place $p0$ and transition $t1$ has not yet been executed. The effect of executing transition $t1$ is to assign F(alse) to $p0$ and T(rue) to $p1$, $p2$, $p3$, and d_{t1}; in symbols, we write $p0, p1, p2, p3, d_{t1} := F, T, T, T, T$. The other transitions are modeled similarly.

Besides the constraints on the execution of tasks, Figure 5.2 shows also the same authorization constraints of Figure 5.1. These are obtained by taking into consideration both the access

Table 5.1: Workflow as symbolic transition system

event	enabled		action	
	CF	Auth	CF	Auth
$t1(u)$	$p0 \wedge \neg d_{t1}$	$a_{t1}(u)$	$p0, p1, p2, p3, d_{t1}$ $:= F, T, T, T, T$	$h_{t1}(u) := T$
$t2(u)$	$p1 \wedge \neg d_{t2}$	$a_{t2}(u) \ \wedge \ \neg \ h_{t3}(u)$ $\wedge \neg \ h_{t1}(u)$	$p1, p4, d_{t2}$ $:= F, T, T$	$h_{t2}(u) := T$
$t3(u)$	$p2 \wedge \neg d_{t3}$	$a_{t3}(u) \wedge \neg h_{t2}(u)$	$p2, p5, d_{t3}$ $:= F, T, T$	$h_{t3}(u) := T$
$t4(u)$	$p3 \wedge \neg d_{t4}$	$a_{t4}(u) \wedge \neg h_{t1}(u)$	$p3, p6, d_{t4}$ $:= F, T, T$	$h_{t4}(u) := T$
$t5(u)$	$p4 \wedge p5 \wedge p6 \wedge$ $\neg d_{t5}$	$a_{t5}(u) \ \wedge \ \neg \ h_{t3}(u) \ \wedge \ \neg \ h_{t2}(u)$	$p4, p5, p6, p7, d_{t5}$ $:= F, F, F, T, T$	$h_{t5}(u) := T$

control policy P granting or denying users the right to execute tasks and the SoD constraints between pairs of tasks. To formalize these, we introduce two functions a_t and h_t from users to Boolean, for each task t, which are such that $a_t(u)$ is true iff u has the right to execute t according to the policy P and $h_t(u)$ is true iff u has executed task t. Notice that a_t is a function that behaves as an abstract interface to the policy P whereas h_t is a function that evolves over time and keeps track of which users have executed which tasks. For instance, the enabling condition for the authorization constraint on task $t1$ is simply $a_{t1}(u)$, i.e. it is required that the user u has the right to execute $t1$, and the effect of its execution is to record that u has executed $t1$, i.e. $h_{t1}(u) := T$ (notice that this assignment leaves unchanged the value returned by h_{t1} for any user u' distinct from u). Notice that it is useless to take into account the SoD constraints between $t1$ and $t2$, $t4$ when executing $t1$ since $t2$ and $t4$ will always be executed afterwards. As another example, let us consider the enabling condition for the authorization constraint on $t2$: besides requiring that u has the right to execute $t2$ (i.e. $a_{t2}(u)$), we also need to require the SoD constraints with $t1$ and $t3$ (not that with $t5$ since this will be executed afterwards), i.e. that u has executed neither $t1$ (i.e. $\neg h_{t1}(u)$) nor $t3$ (i.e. $\neg h_{t3}(u)$). The authorization constraints on the other tasks are modeled in a similar way.

Table 5.1 shows the formalization of all transitions in the extended Petri net of Figure 5.2. The first column reports the name of the transition together with the fact that it is dependent on the user u taking the responsibility of its execution. The second column shows the enabling condition divided in two parts: CF, pertaining to the execution constraints, and Auth, to the authorization constraints. The third and last column list the effects of the execution of the transition again divided in two parts: CF, for the workflow, and Auth, for the authorization.

The initial state of the security-sensitive workflow is described by the *initial* formula I

$$p0 \wedge \bigwedge_{i=1,\ldots,7} \neg pi \wedge \bigwedge_{i=1,\ldots,5} \neg d_{ti} \wedge \bigwedge_{i=1,\ldots,5} \forall u.\neg h_{ti}(u) \tag{5.1}$$

saying that there is just one token in $p0$, no task has been executed, and indeed no user has yet executed any of the tasks, whereas a state of a terminating execution of the workflow by

the *goal* or *final* formula F

$$p7 \wedge \bigwedge_{i=0,\ldots,6} \neg pi \wedge \bigwedge_{i=1,\ldots,5} d_{ti} \tag{5.2}$$

saying that there is just one token in $p7$ and all the tasks have been executed.

Formally, the way in which we specify the transition systems T corresponding to security-sensitive workflows can be seen as an extended version of the assertional framework proposed in [Sha93]. We emphasize that obtaining, from the extended BPM notation of Figure 5.1, the symbolic representation S of the initial and goal formulae with that of the transitions in Table 5.1 is a fully automated process.

Exploring the search space. After obtaining the symbolic representation of the initial and goal states together with the transitions of the security-sensitive workflow, we invoke a symbolic model checker in order to compute the symbolic representation R of the set of (reachable) states visited while executing all possible sequences of transitions leading from an initial to a goal state. A crucial assumption of our approach is that the model checker is able to compute R for any finite number of users. By doing this, the interface functions a_t's can be instantiated with any policy P, i.e. containing any number of users. As a consequence, changes in the authorization policy do not imply to re-run the off-line phase. In summary, our goal is to compute a parametric—in the number n of users—representation of the set of states visited while executing all possible terminating sequences of transitions. From now on, we write R_n to emphasize this fact.

Although the computation of R_n seems to be a daunting task, there exist techniques available in the literature about parameterized model checking (see the seminal paper [ACJT96]) that allow us to do this. Among those available, we have chosen the Model Checking Modulo Theories approach proposed in [GR10a] for its use of first-order formulae as symbolic representation of transition systems and the availability of tools, such as MCMT [GR10b], which are capable of returning the set of reachable states as a first-order formula.

For instance, Figure 5.3 shows a graph-like representation of the formula R_n for the security-

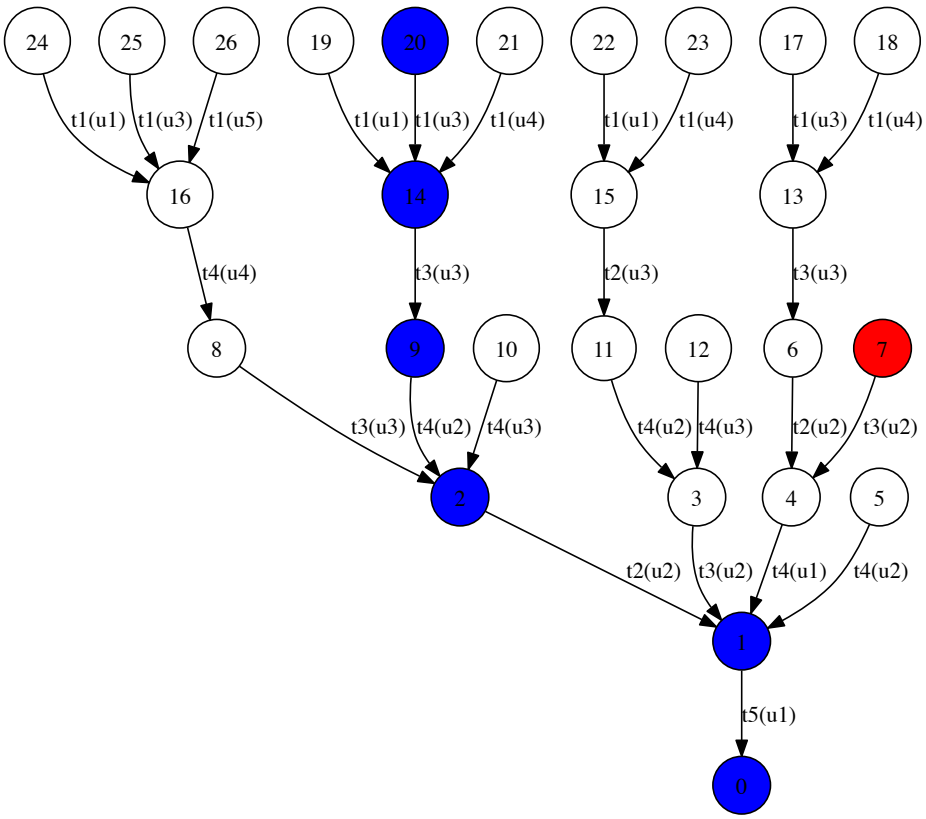


Figure 5.3: Graph-like representation of the set of reachable states for the workflow in Figure 5.1

sensitive workflow described by the symbolic transition system derived from Figure 5.1. Each node is associated to a first-order formula: node 0 (bottom of the figure) is labeled by the goal formula (5.2), nodes 17–26 (top of the figure) are labeled by formulae describing sets of states that have non-empty intersection with the set of initial states characterized by the initial formula (5.1), all other nodes (namely, those from 1 to 16) are labeled with formulae describing sets of states that are visited by executing transitions (labeling the arcs of the graph) belonging to a terminating sequence of executions of the workflow. For instance, node 1 is labeled by the formula

$$\begin{aligned} \neg p0 \wedge \neg p1 \wedge \neg p2 \wedge \neg p3 \wedge p4 \wedge p5 \wedge p6 \quad & \wedge \\ d_{t1} \wedge d_{t2} \wedge d_{t3} \wedge d_{t4} \wedge \neg d_{t5} \quad & \wedge \\ (a_{t5}(u1) \wedge \neg h_{t2}(u1) \wedge \neg h_{t3}(u1)) & \end{aligned}$$

describing the set of states from which it is possible to reach a goal state when some user $u1$ takes the responsibility to execute task $t5$. The first two lines in the formula above require that there is a token in places $p4$, $p5$, $p6$ (thereby enabling transition $t5$), tasks $t1$, $t2$, $t3$, $t4$ have been executed, and $t5$ has not yet been performed. The last line requires that user $u1$ has the right to execute $t5$ and that he/she has performed neither $t2$ nor $t3$ (because of the SoD constraints between $t5$ and $t2$ or $t3$). In general, let us consider an arc $\nu \xrightarrow{t(u)} \nu'$ in the graph of Figure 5.3: the formula labeling node ν describes the set of states from which it is possible to reach the set of states described by the formula labeling node ν' when user u executes task t. Thus, the paths starting from one of the nodes 17–26 (labeled by formulae representing states with non-empty intersection with the set of initial states) and ending in node 0 (labeled by the goal formula) describe all possible terminating executions of the workflow in Figure 5.1 (although nodes 5, 7, 10 and 12 seem to be exceptions, this is not the case: explaining their role requires a more precise description of how the graph is built and will be discussed in the next section). For instance, the sequence of blue nodes describes the terminating sequence $t1, t3, t4, t2, t5$ of task executions by the users $u3$, $u3$, $u2$, $u2$, and $u1$, respectively. It is easy to check that this sequence satisfies both the execution and the authorization constraints required by the workflow in BPM notation of Figure 5.1. In fact, $t1$ is executed first, $t5$ is executed last, and $t2, t3, t4$ are executed in between; there are three *distinct* users $u1, u2, u3$ that can execute the five tasks without violating any of the SoD constraints. By considering all possible paths in the graph of Figure 5.3, it is easy to see that there should be at least three distinct users to be able to terminate the security-sensitive workflow in Figure 5.1. From what we said above, the formula R_n representing the set of states visited during terminating sequences of task executions of the security-sensitive workflow in Figure 5.1 can be obtained by taking the disjunction of the formulae labeling the nodes in the graph of Figure 5.3 except for the one labeling node 0 since, by construction, no task is enabled in the set of states represented by that formula. Let r_ν be the formula labeling node ν, then

$$R_n := \bigvee_{\nu \in N} r_\nu \tag{5.3}$$

where N is the set of nodes in the graph (in the case of Figure 5.3, we have $N = \{1, ..., 26\}$).

In [BdSR15] we have defined an algorithm to compute the set of all possible reachable states of a symbolic security-sensitive workflow. The algorithm takes as input the symbolic security-sensitive workflow S together with the state formula F defining the set of final states and returns a labeled graph RG, called *reachability graph*, whose set of labeled paths is the set of all transitions of S ending with F. The procedure incrementally builds the reachability graph RG by updating the set N of nodes, the set E of edges, and a labeling function λ from N to state formulae (see [BdSR15] for more details and a proof of correctness of the algorithm).

This result implies that starting from an initial state (i.e. one satisfying the initial formula I) in the reachability graph, it is always possible to reach a final state (i.e. one satisfying the final formula F). If no event can be enabled infinitely often without being executed—called *strong fairness*—then a final state is eventually reached. As observed in [vdAvHtH$^+$11], the assumption of strong fairness is reasonable in the context of workflow management since decisions to execute tasks are under the responsibility of applications or humans.

This implies that, if S is in state s and we want to know if a certain user u_0 can execute task t_0 while guaranteeing that the authorization constraints are satisfied and the workflow terminates, it is sufficient to find a node of the reachability graph that is satisfied by s and having one of the outgoing edges labeled by t_0. Indeed, this is exactly the task a monitor is supposed to perform.

5.3.2 On-line phase

Once MCMT has returned the first-order formula R_n describing the set of states visited during any terminating executions for a (finite but unknown) number n of users, we can derive a Datalog [CGT89] program which constitutes the run-time monitor of the security-sensitive workflow formalized by the symbolic transition system used to compute R_n. Then, we can add the specification of the interface functions a_{t1}, ..., a_{t5} for a given value of n.

We have chosen Datalog as the programming paradigm in which to encode monitors for three main reasons. First, it is well-known [LM03] that a wide variety of access control policies can

be easily expressed in Datalog. Second, Datalog permits efficient computations: the class of Datalog programs resulting from translating formulae R_n permits to answer queries in *LogSpace* (see below for more details). Third, it is possible to further translate the class of Datalog programs we produce to SQL statements so that run-time monitors can be easily implemented as database-backed applications. In the rest of this section, we describe how it is possible to derive Datalog programs from formulae describing the set of reachable states computed by the model checker and then how to add the definitions of the interface functions a_{t1}, ..., a_{t5}.

From R_n to Datalog. Recall the form (5.3) of R_n. It is not difficult to see that each r_ν can be seen as the conjunction of a formula r_ν^{CF} containing the Boolean functions $p0, ..., p7$ for places and $d_{t1}, ..., d_{t5}$ keeping track of task execution with a formula r_ν^{Auth} of the form

$$a_t(u0) \wedge \rho_\nu^{\mathrm{Auth}}(u0, u1, ..., uk)$$

where $u0$ identifies the user taking the responsibility to execute task t, ρ_ν^{Auth} is a formula containing the variables $u0, u1, ..., uk$, the interface functions $a_{t1}, ..., a_{t5}$, the history functions $h_{t1}, ..., h_{t5}$, and all disequalities between pairwise distinct variables from $u0, u1, ..., uk$ (indeed, if there are no variables, there is no need to add such disequalities). For instance, formula r_1 labeling node 1 in Figure 5.3 is $r_1^{\mathrm{CF}} \wedge r_1^{\mathrm{Auth}}$ where

$$\begin{aligned} r_1^{\mathrm{CF}} &:= \neg p0 \wedge \neg p1 \wedge \neg p2 \wedge \neg p3 \wedge p4 \wedge p5 \wedge p6 \wedge \\ &\quad\ d_{t1} \wedge d_{t2} \wedge d_{t3} \wedge d_{t4} \wedge \neg d_{t5} \\ r_1^{\mathrm{Auth}} &:= \rho_1^{\mathrm{Auth}}(u1) \\ \rho_\nu^{\mathrm{Auth}}(u1) &:= a_{t5}(u1) \wedge \neg h_{t2}(u1) \wedge \neg h_{t3}(u1) \end{aligned}$$

with $u0$ renamed to $u1$.

In general, each r_ν in the expression (5.3) for the formula R_n can be written as

$$r_\nu^{\mathrm{CF}} \wedge a_t(u0) \wedge \rho_\nu^{\mathrm{Auth}}(u0, u1, ..., uk) \tag{5.4}$$

and describes a set of states in which user $u0$ executes task t while guaranteeing that the workflow will terminate since ν is one of the nodes in the graph computed by the model checker while generating all terminating sequences of tasks. In other words, (5.4) implies that $u0$ can execute task t or, equivalently written as a Datalog clause: $can\_do(u0, t) \leftarrow (5.4)$, where can_do is a Boolean function returning true iff a user (first argument) is entitled to execute a task (second argument) while all execution and authorization constraints are satisfied and the workflow can terminate. Notice that $can\_do(u0, t) \leftarrow (5.4)$ is a Datalog clause. So, we generate the following Datalog clauses

$$can\_do(u0, t) \leftarrow r_\nu^{\mathrm{CF}} \wedge a_t(u0) \wedge \rho_\nu^{\mathrm{Auth}}(u0, u1, ..., uk) \tag{5.5}$$

for each $\nu \in N$. In the following, let D_n be the Datalog program composed of all the clauses of the form (5.5). For instance, the Datalog clause corresponding to node 1 is

$$\begin{aligned} can\_do(u1, t5) \quad \leftarrow \quad & \neg p0 \wedge \neg p1 \wedge \neg p2 \wedge \neg p3 \wedge p4 \wedge p5 \wedge p6 \wedge \\ & d_{t1} \wedge d_{t2} \wedge d_{t3} \wedge d_{t4} \wedge \neg d_{t5} \wedge \\ & a_{t5}(u1) \wedge \neg h_{t2}(u1) \wedge \neg h_{t3}(u1)\,. \end{aligned}$$

It is not difficult to show that $can\_do(u, t)$ iff there exists a disjunct of the form (5.4) in R_n for a given number n of users. Finally, observe that clauses of the form (5.5) contain negations but are non-recursive.

Specifying the policy P. We are left with the problem of specifying the access control policy P for a given number n of users. As already observed above, there should be at least three distinct users in the system to be able to terminate the execution of the workflow in Figure 5.1. So, to illustrate, let $U = \{a, b, c\}$ be the set of users and use the RBAC model to express the policy. This means that we have a set $R = \{r_1, r_2, r_3\}$ of roles which are indirections between users and (permissions to execute) tasks. Let $UA = \{(a, r1), (a, r2), (a, r3), (b, r2), (b, r3), (c, r2)\}$ be the user-role assignments and $TA = \{(r_3, t1), (r_2, t2), (r_2, t3), (r_1, t4), (r_2, t5)\}$ be the role-task assignment. Then, a user u can execute task t iff there exists a role r such that

$(u, r) \in UA$ and $(r, t) \in TA$. This can be formalized by the following Datalog clauses:

$$ua(a, r1)\ ua(a, r2)\ ua(a, r3)\ ua(b, r2)\ ua(b, r3)\ ua(c, r2)$$
$$pa(r_3, t1)\ pa(r_2, t2)\ pa(r_2, t3)\ pa(r_1, t4)\ pa(r_2, t5)$$
$$a_t(u) \leftarrow ua(u, r) \wedge pa(r, t) \text{ for each } t \in \{t1, ..., t5\}$$

and denoted by D_P. By taking the union of the clauses of D_n and D_P, we build a Datalog program $M_{n=3}$ allowing us to monitor the security-sensitive workflow of Figure 5.1, i.e. $M_{n=3}$ is capable of answering queries of the form $can\_do(u, t)$ in such a way that all execution and authorization constraints are satisfied and the workflow execution terminates.

Theorem 5.1 *Let SSW be a security-sensitive workflow and S be the corresponding symbolic transition system. Let R_n be the formula denoting all the reachable states of S as in 5.3. Additionally, let D_P be a Datalog authorization policy over the interface functions a_t, for all tasks t in S. A user $u \in U$ can execute task t guaranteeing the satisfaction of all authorization constraints and the termination of the workflow iff the query $can\_do(u, t)$ is answered positively by the Datalog program $D_n \cup D_P$.*

This result guarantees the correctness of our procedure to synthesize run-time monitors. It is a consequence of the definition of Datalog authorization policy program and the correctness of the off-line procedure. Notice that when both D_n and D_P are non-recursive (stratified) Datalog programs, queries can be answered very efficiently in *LogSpace* and can be translated to SQL without aggregate operators (such as *AVG* and *COUNT*).

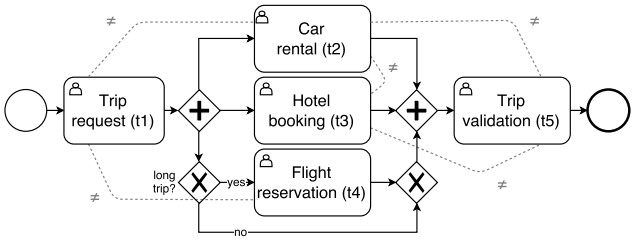
An example of a run of the monitor is in Table 5.2, where each line represents a state of the system; columns CF and Auth describe the values of the variables in that state ("Token in" shows which places have a token and the various h_{ti} hold the name of the user who executed task t_i); $can\_do(u, t)$ represents user u requesting to execute task t and 'Resp' is the corresponding response returned by the monitor (grant or deny the request). The execution in the table shows two denied requests, one in line 0 and one in line 2. In line 0, user a requests to execute task $t1$ but this is not possible since a is the only user authorized to execute $t4$, and if a executes $t1$,

Table 5.2: A run of the monitor program $M_{n=3}$ for the security-sensitive workflow in Figure 5.1

	CF	Auth					*can_do*	
#	Token in	h_{t1}	h_{t2}	h_{t3}	h_{t4}	h_{t5}	(u,t)	Resp.
0	$p0$	-	-	-	-	-	$(a,t1)$	deny
1	$p0$	-	-	-	-	-	$(b,t1)$	grant
2	$p1, p2, p3$	b	-	-	-	-	$(b,t2)$	deny
3	$p1, p2, p3$	b	-	-	-	-	$(a,t2)$	grant
4	$p4, p2, p3$	b	a	-	-	-	$(c,t3)$	grant
5	$p4, p5, p3$	b	a	c	-	-	$(a,t4)$	grant
6	$p4, p5, p6$	b	a	c	a	-	$(b,t5)$	grant
7	$p7$	b	a	c	a	b	-	-

he/she will not be allowed to execute $t4$ because of the SoD constraint between $t1$ and $t4$ (see Figure 5.1). In line 2, user b requests to execute task $t2$ but again this is not possible since b has already executed task $t1$ and this would violate the SoD constraint between $t1$ and $t2$. All the other requests are granted, as they do not violate execution or authorization constraints.

So far, we have described the key ideas underlying our technique while neglecting efficiency considerations related to the enumeration of all possible terminating execution sequences of the security-sensitive workflow. If we want our approach to scale up and handle real-world workflows, we have to design suitable heuristics as discussed in [dS17]. Note however that an advantage of our technique is that changes in the policies can be taken into account without re-running the off-line phase since only an abstract interface to policies is required. The interface is refined to the concrete policy only in the on-line phase. An extensive experimental evaluation with an implementation of the technique shows the scalability of our approach on the important class of hierarchic workflows, as detailed in [BdSR15, dS17].



User	Task	User	Task
a	t1	b	t3
b	t1	c	t3
a	t2	a	t4
b	t2	a	t5
c	t2	b	t5
a	t3	c	t5

Figure 5.4: TRW in BPMN with associated authorization policy TA

5.4 Weighted Workflow Satisfiability Problem

Authorization policies and constraints are crucial to ensure the security of workflow systems and to avoid errors and frauds [LRM14], but they may also lead to situations where a workflow instance cannot be completed because no task can be executed without violating either the authorization policy or the constraints. These deadlocks are conflicts between compliance and continuity which may be resolved by administrators granting additional permissions to users (thus hindering compliance) or canceling the execution (precluding continuity). Depending on the scenario, it may be preferable to guarantee either security or continuity. In all cases, it is desirable to have "minimal" (in some sense) violations. The Multi-Objective Workflow Satisfiability Problem (MO-WSP), that we considered in [BdSR18], amounts to strike the best possible trade-off between security and continuity while minimizing the costs of violations to a policy or constraints. The MO-WSP is inspired by the Valued WSP [CGK15] and its generalization, the Bi-Objective WSP [CGKW17]. In our work, we define variations of the MO-WSP and solve them using bounded model checking and optimization modulo theories solving. Our symbolic solution is also able to handle control-flow patterns [vdAtHKB03], such as alternative execution, since we can encode these patterns directly in the transition system used by BMC (instead of splitting a workflow into multiple deterministic instances, as in [CG13a, CGKW17]). The use of off-the-shelf OMT solvers, instead of custom algorithms, provides a uniform toolkit to explore different optimization modes (such as Pareto and those based on linear cost functions), thereby gaining the freedom to evaluate the trade-offs offered by different optimization strategies.

Let us explain our approach on the example in Figure 5.4:

Suppose no user is entitled to execute task $t2$ (they cannot access the TRW for some reason), then the WSP for the TRW is no more solvable. For the sake of business continuity, it would be important to understand which users can execute task $t2$ (despite none being entitled to do so) to ensure termination while minimizing security issues. This becomes possible as soon as we define the cost of violating an authorization policy and, in the general case, an authorization constraint.

Following [CGK15], we introduce a cost function w_P such that $w_P(u,t) = 1$ if $(u,t) \notin TA$ and $w_P(u,t) = 0$ if $(u,t) \in TA$. The idea is to associate a cost of one to the situation in which a user executes tasks which they are not entitled to execute according to the policy TA; instead, if users are entitled to execute tasks, then the cost is zero since there is no violation of the policy TA. To measure the cost of the violations to the policy TA over the execution of an entire scenario, a possibility is to take the sum of the costs of each violation (if any). In the same spirit, we can introduce an additional cost function w_C for the authorization constraints in K such that $w_C(\eta)$ is equal to the cardinality of the set $\{k \in K \mid \eta \text{ does not satisfy } k\}$ where K contains the SoD constraints of Figure 5.1. Intuitively, w_C counts how many authorization constraints are violated by the scenario under consideration. We are then interested to find the scenarios in TRW that minimize both cost functions w_P and w_C. There are several reasonable ways to solve this problem. For instance, one can minimize the combined cost of w_P and w_C (e.g., by taking their sum) or minimize each one of them. In some situations, it may be unclear which solutions to consider as optimal. Consider, for instance, the criterion of minimizing the two cost functions at the same time and the situation in which two scenarios have costs $(1,2)$ and $(2,1)$, respectively. In order to address this kind of questions, we have decided to define a quantitative version of the WSP, by borrowing some notions from the framework of Multi-Objective Optimization (MOO) problems [MA04].

Indeed, the main goal of MOO techniques is to simultaneously optimize a collection of cost functions. In general, for any non-trivial MOO problem, there is no single solution that is simultaneously optimal for every objective. Instead, there may exist (possibly infinitely) many solutions that can be considered equally good, called Pareto optimal.

We can formalize the problem as follows:

Definition 5.3 (Multi-Objective WSP) *Given a SSW* $(W(T,U), TA, K)$ *with functions* w_P *and* w_C *associating scenarios in* $W(T,U)$ *with the costs of violating the authorization policy TA and the authorization constraints in* K*, respectively; the* Multi-Objective WSP *(MO-WSP) amounts to*

$$\underset{\eta}{minimize} \quad (\ w_P(\eta), w_C(\eta) \ ) \quad subject\ to \quad \eta \in \mathcal{S}$$

where $\mathcal{S} \subseteq W(T,U)$ *is the set of* scenarios of interest.

With respect to the Valued WSP [CGK15] and the Bi-Objective WSP [CGKW17], in MO-WSP the set $W(T,U)$ may contain scenarios of various lengths (because of the presence of conditionals) whereas the class of workflows for which the cited WSPs are defined give rise to scenarios with the same length (as they cannot specify conditional executions). Also, the MO-WSP takes into consideration control-flow constraints whereas the others do not. Indeed, a solution to the Valued or Bi-Objective WSP is an optimal plan (a function assigning tasks to users), whereas a solution to the MO-WSP is an optimal execution scenario. In general, there are valid plans which cannot become valid execution scenarios, as observed in [CG13a].

Following our approach, we encode all the possible executions of the considered SSW as a SMT bounded model checking (BMC) problem, with an encoding similar to the one used for the off-line phase described in the previous section. To mechanize this process, we need a SMT solver, or simply a SAT solver after encoding our first-order formula BMC into a purely Boolean formula (which is possible because the set U of users is finite). We also define a symbolic representation of the set S of scenarios of interest, which is conjoint to the BMC formula. It is important to remark that some of the most important control flow patterns in BPMN for parallel and non-deterministic/conditional executions can be expressed in the so-called (1-bounded) *casual* class of Petri nets; see, e.g., [DDO08]. The advantage of considering causal nets is a simplified symbolic representation of the set $\mathcal{S}$ of the scenarios of interest. In fact, the conditionals (such as *long trip*? in our example) can be executed only once. One of the most important omissions is iteration (as considered in, e.g., [CGW17]), which we do not

consider for the time being. To solve our problem, we consider an extension of SMT called OMT (optimization modulo theories) which aims to solve the problem of finding a model for an input formula which is optimal with respect to one or more objective functions. We use our BMC formula, together with the cost functions, as input to an OMT solver. We can choose one optimization mode among the techniques available in the literature (see, e.g., [MA04, ST15, BPF15]), provided that it is implemented in the OMT solver.

We carried on an evaluation of the proposed solution on real and synthetic workflows by using PySMT [GM15], and the solvers OptiMathSAT [ST15] and Z3 [BPF15]. Both natively support Boxed, Lexicographic, and Pareto optimization. The results show that the technique has a good performance due to an ingenious encoding of the problem that exploits the parallel executions of tasks in the workflow. Indeed, by exploiting properties of causal Petri nets, we can obtain the sets of tasks that can be executed in parallel at each transition step by computing the lattice of maximal anti-chains (see, e.g., [Gar13]) and traversing it breadth-first. The number of steps becomes thus equal to the number of maximal anti-chains, instead of equal to the number of tasks in the workflow. After discharging the BMC formula to an OMT solver, the resulting model is a compact representation of several possible interleaving executions, which can be linearized. It is not surprising that this parallel encoding has a superior performance in all the experiments, compared to the standard interleaving encoding. Details about the implementation and scalability results can be found in [BdSR18].

5.5 Related work

Model Checking Modulo Theories [GR10a] is an approach for the verification of array-based systems based on the computation of pre-images of a set of states using first-order formulae and on reducing fix-point checks to SMT solving. This approach is implemented by the model checker MCMT [GR10b]. In [BR13b], we have made the link between array-based systems and security-sensitive workflows explicit by means of what we call *composed* array-based systems. We have also presented a terminating procedure for the verification of reachability properties

for this class of systems.

One of the first works about workflow satisfiability is the one by Bertino et al. in [BFA99] where the authors describe the specification and enforcement of authorization constraints in workflow management systems, presenting constraints as clauses in a logic program and an exponential algorithm for assigning users and roles to tasks without violating them, but considering only linear workflows. Crampton [Cra05] showed another model for specifying constraints, considering workflows as DAGs, and an algorithm to determine whether there is an assignment of users to tasks that satisfies the constraints, showing that it can be incorporated into a reference monitor. [CK08] extended the previous work to consider the effects of delegation on satisfiability, showing algorithms to only allow delegations that can still satisfy a workflow. Crampton et al. [CHK14] used model checking on an NP-complete fragment of linear temporal logic to decide the satisfiability of workflow instances. The authors presented three different encodings in LTL(F) that can compute a set of solutions, minimal user bases and a safe bound on resiliency. The synthesis of monitors was left as future work.

Wang and Li [WL10] proposed a role-and-relation based access control model that allows to describe the relationships between users and thus specify complex authorization constraints. The authors showed that the WSP is NP-complete in their model and that this intractability is inherent in authorization systems supporting simple constraints. They showed that with only equality and inequality relations, using the number of tasks as a parameter renders the WSP fixed-parameter tractable. They also reduced the problem to SAT. Yang et al. [PYL13] studied the complexity of several formulations of the WSP, considering the possibility of different control-flow patterns, and showed that, in general, the problem is intractable.

Basin et al. [BBK12b] considered the problem of choosing authorization policies that allow a successful workflow execution and an optimal balance between system protection and user empowerment. They treat the problem as an optimization problem (finding the cost-minimizing authorization policy that allows a successful workflow execution) and show that, in the role-based case, it is NP-complete. They generalize the decision problem of whether a given authorization policy allows a successful workflow execution to the notion of an optimal authorization

policy that satisfies this property. In a following work, Basin et al. [BBK12a] used the Separation of Duties Algebra (SoDA) to enforce SoD constraints in a dynamic, service-oriented enterprise environment. The authors generalized SoDA's semantics to workflow traces that satisfy a term and refined it for control-flow and role-based authorizations. Their formalization, based on CSP, is the base for provisioning SoD as a Service, with an implementation using a workflow engine and a SoD enforcement monitor. [BBK12a] is the closest to us in terms of monitor implementation, but their monitor only verifies if a trace of a workflow satisfies a SoDA term, being incapable of checking whether there is a future trace that can be concatenated in order to satisfy the workflow.

On the other hand, Crampton et al. [CGK15, CGKW17] first studied how to find minimal violating assignments of users to tasks, without changing the policy. They first defined the Valued WSP [CGK15] and later the BO-WSP [CGKW17], then solved both using a bespoke algorithm and showed that their solution is superior to a mixed integer programming approach in terms of performance. The authors also showed how to solve two related problems by encoding them as cost functions: the quantitative resiliency problem [MMM14], which amounts to assigning a probability to the successful termination of a workflow even in the absence of some users; and the Cardinality-constrained Minimum User Problem (CMUP) [RSM$^+$15], which consists in finding the minimum number of users required to satisfy a workflow instance. The main difference between our work and Crampton et al. [CGK15, CGKW17] is that we consider an ordered execution of workflows, whereas they take as solution a valid plan, which is an unordered assignment of tasks to users. They also considered user-independent constraints in their experiments, which we did not implement, but can be expressed in the fragment of first-order logic that we use.

Crampton et al. [CGW17] extended their algorithmic solution to support conditional workflows with release points—which specify that a constraint may be active only for some scenarios—by splitting a workflow instance into many deterministic ones. We believe that release points can also be incorporated in our solution by using an approach similar to [BBK12b]. A challenge is to adapt the parallel encoding we define to tackle this generalized problem so to have better scalability. For this, we believe that the techniques in [DJH12] can be useful.

For a more complete survey on workflow satisfiability approaches the reader can refer to [HAM15, dSR17].

Chapter 6

Conclusion

6.1 Summary and perspectives

We detail in the following the contributions and perspectives corresponding to each chapter of this thesis.

Chapter 3 We have described a declarative framework for the specification and analysis of distributed access control policies based on the metamodel called CBAC. We have given rewrite-based methodology that can be applied to the verification of security properties such as consistency and totality for security policies specified in our framework. Moreover, we have used narrowing techniques to make change-impact analysis of dynamic policies, as a consequence of administrative actions. We think narrowing can contribute to improve the trust of an administrator with respect to a policy, not only at the design stage, but also whenever a policy is updated. A further development to this aim would be to consider answering general queries over rewrite-based policies. For instance, one would ask whether access is always denied for a given resource. Since narrowing can tell the possible instantiations for a term that will fire some rule, this may help answer questions of the kind "what rules apply in this situation" or "what-if" questions, like "what if a given user is assigned an additional category", also known as "administrator queries" [JSSS01, KKdO09].

The specification of methods to check equivalence, permissiveness and definedness of CBAC policies would be useful for security administrators. In addition to being useful for conflict detection, policy comparison methods and algorithms can be used to detect redundancies and inconsistencies in policies. If two policies are equivalent, they can be used to simplify the policy specification. For CBAC policies, equivalence can be checked by analysing the category definitions. CBAC relies on a core axiom to generate policy decisions, based on the categorisation of principals and resources. Equivalence of categories is undecidable in general, since categories can be defined for example via a program and program equivalence is undecidable in general. However, in standard cases (e.g., the RBAC instance of CBAC or simple ABAC instances where attributes have finite domains) policy equivalence is decidable. If the policies are not equivalent, a disagreement set can be computed and we can establish an order between policies (such as more permissive policy, or more defined policy [BDH07]).

Term rewriting rules provide an executable specification of the access control policy. A first-order rewriting system can be transformed into a MAUDE program simply by adding type declarations for the function symbols and variables used and by making minor syntactical changes [CDE+03, San08]. Policies defined by higher-order rewriting systems can be directly implemented in a functional language [BF08c]. The next step for catgeory-based policies will be their practical implementation. It would be interesting to see how aspect-oriented techniques apply to this case (work in this direction has been reported in [dOWKK07], where the authors discuss weaving rewrite-based policies into Java programs). In our view, a comprehensive tool integrating policy specification, administration and analysis exploiting existing rewrite-based languages and tools (such as [CPU+10, GSKT06, CDE+03] could be of great help to policy designers and administrators in their policy definition and management tasks. We recently made a first step in this direction by developing a preliminary prototype with a graphical interface for defining and analysing CBAC policies [1].

Chapter 4 We have proposed a framework for the specification and enforcement of access control in presence of transitive dependencies for entreprise portals as well as an implementation

[1] Prototype available at: https://projectadmincbac.000webhostapp.com

based on the standard XACML architecture extended with a delegation module.

In our work we assume that the topology of the system (i.e. how the integrated services may invoke each other) is known and fixed. It would be interesting to investigate how to lift our techniques to SOA applications with dynamic topologies, which are particularly useful in practice to match the rapidly evolving requirements of modern business models. For the performances of the technique at both design- and run-time, it will be critical to identify situations in which authorization requests can be efficiently answered as it is the case with a static topology (recall that answering a query takes polynomial time if we consider only data complexity). Another aspect that has not been explicitly treated is the administration of the security policies. We can imagine to develop an administrative model along the lines of the Ad-OrBAC model [CM04], to control assignments of different entities in our D-OrBAC model. It would also be interesting to consider issues about trust negotiation [LSWY07]. Since interactions occur between entities not sharing the same security domain, trust negotiation would help in the process of defining an agreement between the organisations to converge towards a set of shared delegations.

In Chapter 4 we also considered multi-source cooperative environments and presented an access control policy framework that takes into account the policies associated to the data used in the generation of the requested object. Specifically, our framework extend the ABAC paradigm with the notion of provenance information to keep track of the evolution of objects in the cooperative system. We have illustrated how provenance-based constraints can be implemented in existing XACML-based access control frameworks. In this work, we have considered core provenance information for policy specification. We believe that accounting for contextual information in the provenance graph would enable the definition of more fined-grained policies. A possible extension of our work is thus to enrich our policy language and provenance graphs to accommodate contextual information along the lines of the PROV-DM data model [MM13] where provenance graphs are augmented with attributes in form of relation annotations. The use of a standardized provenance model will allow us to reuse provenance graphs generated by existing systems.

Another interesting line of research is the application of the model to community-centered

collaborative systems. As part of Alba Anton Martinez PhD thesis, we aim to develop a general framework expressive enough to represent cooperative management of multi-owner information and allow to determine whether a data disclosure meets the privacy expectations of the different involved parties. Moreover, it should facilitate transparent access for authorized users and give support for controlled sharing by means of collaborative decision making. The idea is to focus on controlled data sharing by applying different sharing strategies producing access decisions that are more expressive than simple binary policies. As conflicts are inevitable in multi-party access control, a flexible conflict resolution mechanism needs to be addressed to cope with authorization and privacy conflicts.

Chapter 5 We have presented an automated technique for monitoring workflows with authorization constraints, provided an implementation and a thorough experimental evaluation. The main advantages of our work are the specification of the security-sensitive workflows as array-based systems and the consideration of an off-line and an on-line phase. These characteristics allow us to efficiently compute all terminating executions of large instances of workflows for a finite but unbounded number of users and then translate it to a Datalog program that acts as an efficient run-time monitor. We have also addressed the Multi-objective workflow satisfiability problem and proposed a methodology for finding a (pareto set of) solutions optimal with respect to some given cost functions associated to the violation of authorization constraints. This may be useful when a trade off is needed between business continuity and security constraints.

In our work we have considered a few examples of authorization constraints, e.g., SoD and BoD. These are the most common in practice, however in the fragment of first-order logic we consider, other constraints such as counting constraints can be easily specified (see [dS17]). Other, more complex, constraints could also be supported, as for example instance-spanning constraints [LMRM12]. Instance-spanning constraints restrict what users can do across several instances of the same workflow (inter-instance), across several instances of different workflows (inter-process), or across workflows in different organizations (inter-organization). The most usual case is inter-instance authorization constraints, which have been studied in, e.g., [WA06].

Since in our approach we have one monitor for each instance, support for inter-instance constraints would require a global synchronization of the states of each monitor, possibly using a global execution history stored in a central entity to which each monitor communicates. In this configuration, each monitor should ask the decision of the central entity before taking the right decision to avoid the violation of some inter-instance constraint.

Besides the workflow satisfiability problem, other related problems have been studied in the literature. Workflow Resiliency for instance [WL10] amounts to checking if a workflow can still be satisfied even in the absence of a certain number of users, while Workflow Feasibility [KF12] concerns the question of whether there is a possible configuration of the authorization policy in which the workflow is satisfiable. In [dS17], our solution to WSP is adapted to deal with resiliency by refining the reachability graph computed by the model checker with a given authorization policy. This approach is inspired by the solution to the Resiliency Checking Problem in [CGW16]. Compared to other works, our technique has the advantage of reusing the heaviest part of the computation (i.e., generating the reachability graph) and only refining it during deployment. The algorithm presented as a solution is also capable of finding execution scenarios that satisfy other constraints, such as a particular user executing a task or using a minimal number of users. We think the solution can be adapted for quantitative resiliency (i.e. how likely a workflow instance is to terminate given a user availability model) by assigning weights representing availability to the edges in the reachability graph, as hinted at in [CGW16]. Another interesting related question is, given a workflow specification with costs and a "budget", find all possible workflow instances that have an allocation of users to steps not exceeding the budget [CGM19].

6.2 Discussion and future challenges

The use of formal methods in many computer science disciplines has improved the quality and reliability of computer systems. Formal methods have been successfully used to ensure security properties of e.g., hardware and network architectures, web applications and services, software

systems [GdHN+08, NRZ+15, RFR+12, TWM+09, KKP+15, FCKV10].

However, as reported in a recent NSF workshop on formal methods for security [CGD+16], significant challenges must still be overcome. Nowadays, the increasing complexity of our socio-economic environment produces large distributed systems, with data storage and computation happening at different sites. These new architectures (cloud computing, Software as a Service or System of Systems) include new features, such as reconfigurable settings, dynamism of the environment, evolution of uses, and make new security properties or constraints arise. A major challenge in the next years will be to make the methods of verification evolve to take into account these new architectures and the new associated properties, by developing for instance modular or parametric verification. A first serious difficulty lies in the fact that security requirements of distributed systems are hard to specify and formalize. Moreover, distributed systems security is a big problem that involves several aspects and expertise from multiple research areas: operating systems and networking, programming languages, cryptography, etc. A real progress on verification of distributed systems security could come from the combination of the research efforts of all these research communities.

From a user perspective, in this interconnected, dynamic environment, where components and users may evolve at any time, another important issue concerns privacy-relevant guarantees. It is important to give users the ability of understanding how their data are managed and communicated to third parties, as well as the possibility of personalizing their privacy settings. This is essential for increasing user awareness about security problems. Users frequently do not understand what a security policy really checks, and hence are unaware of the risks involved in many common operations. Most users have no specific training in programming nor in formal logics. We believe adopting a rule-based language is a first step for greater user awareness and control on policies, since it is flexible enough and at the same time structurally similar to the way in which policies are expressed by nontechnical users. In addition, it is necessary to provide user-friendly front-ends that illustrate the policy in a language familiar to the user, such as a graphical language. In [MS18] for instance the Javascript tool VisABAC for authoring and editing access control policies is presented. VisABAC proposes the graphical representation of attribute based access control polices using visualisation techniques (diagrams, colour schemes,

patterns...). Some other research works [BDF+06, MP04] develop explanation mechanisms in the context of (Semantic-) Web based systems to help the user understand what policies prescribe and control.

All these issues are related to the more general notion of *explainability* of a decision making system. Nowadays, many decision processes that were taken by humans can be performed with the assistance of, or directly by, an algorithmic autonomous process. This is true in many contexts, from finance to healtcare, to smart home environments. Decision making system are often complex systems that are difficult to understand, especially when based on machine learning algorithms. Be able of explaining why the system arrived at a specific decision becomes thus a crucial issue. The explanation may be of different nature (operational, logical, causal,...) One of the options to achieve explainability is the constructive approach, where explainability requirements are taken into account by design. The quality of explanations is also something that needs to be considered. So far, there is no systematic method or set of benchmarks allowing a precise evaluation of explainability. Usually evaluation is carried on by means of human experiments (e.g. to assess what kinds of explanations are better understood) or field experiments with the users of the systems. Evaluations that are based on formal definitions (for example using decision trees or decision rules as explainable models) are less expensive because they do not require the setting up of experiments but rely on assumptions (the formal definitions) which have already been validated. A trade-off needs to be found among accuracy, cost and explainability. The implementation of explainability by design is an open question, but we believe that rule-based formal specification may help in defining self-explainable policies. Still, an explanation for a whole process, usually obtained by combining several component modules, remains challenging as independence and coordination of the modules has to be taken into account.

Bibliography

[AA20] Mehdi Adda and Linda Aliane. Hobac: fundamentals, principles, and policies. *Journal of Ambient Intelligence and Humanized Computing*, 11(12):5927–5941, 2020.

[Aba03] M. Abadi. Logic in access control. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 228–233, 2003.

[ABLP93] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. 15(4):706–734, 1993.

[ACCP12] A. Armando, R. Carbone, L. Compagna, and G. Pellegrino. Automatic Security Analysis of SAML-Based Single Sign-On Protocols. In Raj Sharman, Sanjukta Das Smith, and Manish Gupta, editors, *Digital Identity and Access Management: Technologies and Frameworks*, pages 168–187. IGI Global, Hershey, PA, USA, 2012.

[ACJT96] P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite-state systems. In *Proc. of LICS*, pages 313–321, 1996.

[ADS04] C.A. Ardagna, E. Damiani S. De Capitani di Vimercati, and P. Samarati. Xml-based access control languages. *Information Security Technical Report*, 9(3):35–46, July-September 2004.

[AF17] S. Alves and M. Fernández. A graph-based framework for the analysis of access control policies. *Theor. Comput. Sci.*, 685:3–22, 2017.

[AG00] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1):133 – 178, 2000.

[AMC+11] Peter Alvaro, William R. Marczak, Neil Conway, Joseph M. Hellerstein, David Maier, and Russell Sears. Dedalus: Datalog in time and space. In Oege de Moor, Georg Gottlob, Tim Furche, and Andrew Sellers, editors, *Datalog Reloaded*, pages 262–281, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[AP10] Alessandro Armando and Serena Elisa Ponta. Model checking of security-sensitive business processes. In Pierpaolo Degano and Joshua D. Guttman, editors, *Formal Aspects in Security and Trust*, pages 66–80, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[AR12] Alessandro Armando and Silvio Ranise. Scalable automated symbolic analysis of administrative role-based access control policies by smt solving. *J. Comput. Secur.*, 20(4):309–352, July 2012.

[AS17] T. Ahmed and R. Sandhu. Safety of ABACα is decidable. In *Network and System Security - 11th International Conference, NSS 2017, Helsinki, Finland, August 21-23, 2017, Proceedings*, pages 257–272, 2017.

[BAB+07] Agreiter Berthold, Muhammad Alam, Ruth Breu, Michael Hafner, Alexander Pretschner, Jean-Pierre Seifert, and Xinwen Zhang. A technical architecture for enforcing usage control requirements in service-oriented architectures. In *Proceedings of the 2007 ACM Workshop on Secure Web Services*, SWS '07, pages 18–25, New York, NY, USA, 2007. ACM.

[Bar09] Steve Barker. The next 700 access control models or a unifying meta-model? In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, SACMAT '09, page 187–196, New York, NY, USA, 2009. Association for Computing Machinery.

[BBG06] R. Bhatti, E. Bertino, and A. Ghafoor. X-FEDERATE: A policy engineering framework for federated access management. *IEEE Transactions on Software Engineering*, 32(5):330–346, 2006.

[BBG07] R. Bhatti, E. Bertino, and A. Ghafoor. An Integrated Approach to Federated Identity and Privilege Management in Open Systems. *Comm. of the ACM*, 50(2):81–87, 2007.

[BBGG09] Steve Barker, Guido Boella, Dov M. Gabbay, and Valerio Genovese. A meta-model of access control in a fibred security language. *Studia Logica*, 92(3):437–477, 2009.

[BBK11] D. Basin, S. J. Burri, and G. Karjoth. Obstruction-free authorization enforcement: Aligning security with business objectives. In *2011 IEEE 24th Computer Security Foundations Symposium*, pages 99–113, 2011.

[BBK12a] David Basin, Samuel J. Burri, and Günter Karjoth. Dynamic enforcement of abstract separation of duty constraints. *ACM TISSeC*, 15(3):13:1–13:30, November 2012.

[BBK12b] David Basin, Samuel J. Burri, and Günter Karjoth. Optimal workflow-aware authorizations. In *Proc. of SACMAT '12*, pages 93–102, New York, NY, 2012. ACM.

[BBU17] Clara Bertolissi, Omar Boucelma, and Worachet Uttha. Enhancing security in the cloud: When traceability meets access control. In *12th International Conference for Internet Technology and Secured Transactions, ICITST 2017, Cambridge, United Kingdom, December 11-14, 2017*, pages 365–366. IEEE, 2017.

[BCDP05] Elisa Bertino, Barbara Catania, Maria Luisa Damiani, and Paolo Perlasca. Geo-rbac: A spatially aware rbac. In *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*, SACMAT '05, pages 29–37, New York, NY, USA, 2005. ACM.

[BCFP03] Elisa Bertino, Barbara Catania, Elena Ferrari, and Paolo Perlasca. A logical framework for reasoning about access control models. *ACM Trans. Inf. Syst. Secur.*, 6(1):71–127, 2003.

[BD04] S. Barker and P. Douglas. Protecting federated databases using a practical implementation of a formal RBAC policy. In *Int. Conf. on Information Technology: Coding and Computing*, volume 1, pages 523–527. IEEE Computer Society, Washington DC, USA, 2004.

[BdCdVS00] P. Bonatti, S. de Capitani di Vimercati, and P. Samarati. A modular approach to composing access control policies. In *CCS'00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 164–173, New York, NY, USA, 2000. ACM Press.

[BDF05] Massimo Bartoletti, Pierpaolo Degano, and Gian Luigi Ferrari. History-based access control with local policies. In *Proceedings of International Conference on Foundations of Software Science and Computation Structures*, pages 316–332. Springer, 2005.

[BDF+06] P. A. Bonatti, C. Duma, N. Fuchs, W. Nejdl, D. Olmedilla, J. Peer, and N. Shahmehri. Semantic web policies – a discussion of requirements and research issues. In York Sure and John Domingue, editors, *The Semantic Web: Research and Applications*, pages 712–724, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[BDH07] Glenn Bruns, Daniel S Dantas, and Michael Huth. A simple and expressive semantic framework for policy composition in access control. In *Proceedings of the 2007 ACM Workshop on Formal Methods in Security Engineering*, FMSE '07, page 12–21, New York, NY, USA, 2007. Association for Computing Machinery.

[BDHdS02] G. Barthe, G. Dufay, M. Huisman, and S. Melo de Sousa. Jakarta: a toolset to reason about the JavaCard platform. In *Proceedings of e-SMART'01*, number 2140 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

[BdHZ19] Clara Bertolissi, Jerry den Hartog, and Nicola Zannone. Using provenance for secure data fusion in cooperative systems. In *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*, SACMAT '19, page 185–194, New York, NY, USA, 2019. Association for Computing Machinery.

[BdSR15] Clara Bertolissi, Daniel Ricardo dos Santos, and Silvio Ranise. Automated synthesis of run-time monitors to enforce authorization policies in business processes. In Feng Bao, Steven Miller, Jianying Zhou, and Gail-Joon Ahn, editors, *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015*, pages 297–308. ACM, 2015.

[BdSR18] Clara Bertolissi, Daniel R. dos Santos, and Silvio Ranise. Solving multi-objective workflow satisfiability problems with optimization modulo theories techniques. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*, SACMAT '18, page 117–128. Association for Computing Machinery, 2018.

[BF06] S. Barker and M. Fernández. Term rewriting for access control. In *Data and Applications Security. Proceedings of DBSec'2006*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[BF08a] C. Bertolissi and M. Fernández. An algebraic-functional framework for distributed access control. In *Proceedings of CRISIS 2008, 3rd International Conference on Risk and Security of Internet System, Tozeur, Tunisia, 2008.*, IEEExplore. IEEE, 2008.

[BF08b] C. Bertolissi and M. Fernández. A rewriting framework for the composition of access control policies. In *Proceedings of the 10th ACM-SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP'08), Valencia, 2008.* ACM Press, 2008.

[BF08c] C. Bertolissi and M. Fernández. Time and location based services with access control. In *NTMS 2008, 2nd International Conference on New Technologies, Mobility and Security, 2008, Tangier, Morocco*, pages 1–6. IEEE, 2008.

[BF09] C. Bertolissi and M. Fernández. Distributed event-based access control. *International Journal of Information and Computer Security, Special Issue: selected papers from Crisis 2008*, 3(3–4), 2009.

[BF10a] C. Bertolissi and M. Fernández. Category-based authorisation models: operational semantics and expressive power. In *Proc. of Int. Symposium on Engineering Secure Software and Systems, ESSOS 2010, Pisa*, number 5965 in Lecture Notes in Computer Science, pages 140–156. Springer, 2010.

[BF10b] Clara Bertolissi and Maribel Fernandez. Category-based authorisation models: Operational semantics and expressive power. In *Engineering Secure Software and Systems*, volume 5965, pages 140–156. Springer Berlin Heidelberg, 2010.

[BF11] C. Bertolissi and M. Fernández. Rewrite specifications of access control policies in distributed environments. In *Proc. of STM 2010: 6th Workshop on Security and Trust Management, Athens, Greece, 2010*, number 6710 in Lecture Notes in Computer Science. Springer, 2011.

[BF14] Clara Bertolissi and Maribel Fernández. A metamodel of access control for distributed environments. *Inf. Comput.*, 238(C):187–207, 2014.

[BF22] Clara Bertolissi and Maribel Fernández. Modular composition of access control policies: A framework to build multi-site multi-level combinations. In Sven Dietrich, Omar Chowdhury, and Daniel Takabi, editors, *SACMAT '22: The 27th ACM Symposium on Access Control Models and Technologies, New York, NY, USA, June 8 - 10, 2022*, pages 7–18. ACM, 2022.

[BFA99] Elisa Bertino, Elena Ferrari, and Vijay Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, February 1999.

[BFB07] C. Bertolissi, M. Fernández, and S. Barker. Dynamic Event-based Access Control as Term Rewriting. In *In Proc. DBSEC 2007, LNCS*. Springer, 2007.

[BFG10] Moritz Y. Becker, Cédric Fournet, and Andrew D. Gordon. Secpal: Design and semantics of a decentralized authorization language. *J. Comput. Secur.*, 18(4):619–665, 2010.

[BFT20] Clara Bertolissi, Maribel Fernández, and Bhavani Thuraisingham. Admin-cbac: An administration model for category-based access control. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, CODASPY '20, page 73–84, New York, NY, USA, 2020. Association for Computing Machinery.

[BFT21] Clara Bertolissi, Maribel Fernández, and Bhavani Thuraisingham. Graph-based specification of admin-cbac policies. In *Proceedings of the ACM Conference on Data and Application Security and Privacy*, CODASPY '21, page ??–??, New York, NY, USA, 2021. Association for Computing Machinery.

[BGJLR07] Yohan Boichut, Thomas Genet, Thomas P. Jensen, and Luka Le Roux. Rewriting Approximations for Fast Prototyping of Static Analyzers. In *Rewriting Techniques and Applications*, pages 48–62, France, 2007.

[Bha06] R. Bhatti. *A Policy Engineering Framework for Federated Access Management*. PhD thesis, Center for Education and Research in Information Assurance and Security, Purdue University, 2006.

[BJ12] Adel Bouhoula and Florent Jacquemard. Sufficient completeness verification for conditional and constrained trs. *J. of Applied Logic*, 10(1):127–143, 2012.

[Bla16] Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Foundations and Trends in Privacy and Security*, 1(1–2):1–135, 2016.

[BLW05] Lujo Bauer, Jay Ligatti, and David Walker. Composing security policies with polymer. *SIGPLAN Not.*, 40(6):305–314, 2005.

[Bou94] Adel Bouhoula. Spike: A system for sufficient completeness and parameterized inductive proofs. In Alan Bundy, editor, *Automated Deduction — CADE-12*, pages 836–840, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[BPF15] N. Bjørner, A. Phan, and L. Fleckenstein. νz - an optimizing SMT solver. In *Proc. of TACAS*, 2015.

[BR13a] Clara Bertolissi and Silvio Ranise. A methodology to build run-time monitors for security-aware workflows. In *8th International Conference for Internet Technology and Secured Transactions, ICITST 2013, London, United Kingdom, December 9-12, 2013*, pages 501–502. IEEE, 2013.

[BR13b] Clara Bertolissi and Silvio Ranise. Verification of composed array-based systems with applications to security-aware workflows. In Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt, editors, *Frontiers of Combining Systems*, pages 40–55, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Bro08] P.C. Brown. *Implementing SOA: Total Architecture in Practice*. TIBCO Press Series. Addison-Wesley, 2008.

[BS03] P. A. Bonatti and P. Samarati. Logics for authorization and security. In J. Chomicki, R. van der Meyden, and G. Saake, editors, *Logics for Emerging Applications of Databases*, pages 277–323. Springer, 2003.

[BS04] Moritz Y. Becker and P. Sewell. Cassandra: distributed access control policies with tunable expressiveness. *Proceedings. Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, 2004. POLICY 2004.*, pages 159–168, 2004.

[BSPM06] Elisa Bertino, Anna C. Squicciarini, Ivan Paloscia, and Lorenzo Martino. Ws-ac: A fine grained access control system for web services. *World Wide Web*, 9(2):143–171, Jun 2006.

[BTV16] Clara Bertolissi, Jean-Marc Talbot, and Didier Villevalois. Analysis of access control policy updates through narrowing. In *Proceedings of the 18th Inter-*

national Symposium on Principles and Practice of Declarative Programming, PPDP '16, page 62–75. Association for Computing Machinery, 2016.

[BU13a] C. Bertolissi and W. Uttha. Automated analysis of rule-based access control policies. In *Proc. of the 6th workshop on Programming Languages meet Program Verification (PLPV'13) affiliated with POPL'13, Rome, Italy, January 22, 2013*. ACM, 2013.

[BU13b] C. Bertolissi and W. Uttha. Automated Analysis of Rule-based Access Control Policies. In *Proceedings of the 7th Workshop on Programming Languages Meets Program Verification*, PLPV '13, pages 47–56, New York, NY, USA, 2013. ACM.

[BVS02] P. Bonatti, S. Vimercati, and P. Samarati. An algebra for Composing access control policies. *TISSEC*, 5(1):1–35, 2002.

[Car14] Pierre Carbonnelle. *pyDatalog*, 2014.

[Cas97] Silvana Castano. *An Approach To Deriving Global Authorizations in Federated Database Systems*, pages 58–75. Springer US, Boston, MA, 1997.

[CB04] B. N. Chun and A. Bavier. Decentralized trust management and accountability in federated systems. In *37th Annual Hawaii Int. Conf. on System Sciences*. IEEE Computer Society, Washington DC, USA, 2004.

[CC97] Laurence Cholvy and Frederic Cuppens. Analyzing consistency of security policies. In *1997 IEEE Symposium on Security and Privacy*, pages 103–112, Oakland, CA, 1997. IEEE Computer Society Press.

[CCBG07] Frédéric Cuppens, Nora Cuppens-Boulahia, and Meriam Ben Ghorbel. High level conflict management strategies in advanced access control models. *Electronic Notes in Theoretical Computer Science*, 186:3 – 26, 2007. Proceedings of the First Workshop in Information and Computer Security (ICS 2006).

[CDE+03] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The Maude 2.0 system. In *Rewriting Techniques and Applications (RTA 2003)*, number 2706 in Lecture Notes in Computer Science, pages 76–87. Springer-Verlag, 2003.

[CF18] Pietro Colombo and Elena Ferrari. Access Control in the Era of Big Data: State of the Art and Research Directions. In *Proceedings of Symposium on Access Control Models and Technologies*, pages 185–192. ACM, 2018.

[CG13a] J. Crampton and G. Gutin. Constraint expressions and workflow satisfiability. In *Proc. of SACMAT*, 2013.

[CG13b] Jason Crampton and Gregory Gutin. Constraint expressions and workflow satisfiability. In *Proceedings of the 18th ACM Symposium on Access Control Models and Technologies*, SACMAT '13, page 73–84, New York, NY, USA, 2013. Association for Computing Machinery.

[CGD+16] Stephen Chong, Joshua Guttman, Anupam Datta, Andrew C. Myers, Benjamin C. Pierce, Patrick Schaumont, Tim Sherwood, and Nickolai Zeldovich. Report on the NSF workshop on formal methods for security. *CoRR*, abs/1608.00678, 2016.

[CGGJ16] Jason Crampton, Andrei V. Gagarin, Gregory Z. Gutin, and Mark Jones. On the workflow satisfiability problem with class-independent constraints for hierarchical organizations. *TOPS*, 19(3):1–29, 2016.

[CGK15] J. Crampton, G. Gutin, and D. Karapetyan. Valued workflow satisfiability problem. In *Proc. of SACMAT*, 2015.

[CGKW17] Jason Crampton, Gregory Z. Gutin, Daniel Karapetyan, and Rémi Watrigant. The bi-objective workflow satisfiability problem and workflow resiliency. *Journal of Computer Security*, 25:83–115, 2017.

[CGM19] Jason Crampton, Gregory Z. Gutin, and Diptapriyo Majumdar. Bounded and approximate strong satisfiability in workflows. In *Proceedings of the 24th*

ACM Symposium on Access Control Models and Technologies, SACMAT 2019, Toronto, ON, Canada, June 03-06, 2019, pages 179–184. ACM, 2019.

[CGT89] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *Knowledge and Data Engineering, IEEE Transactions on*, 1(1):146–166, Mar 1989.

[CGW16] Jason Crampton, Gregory Gutin, and Rémi Watrigant. Resiliency policies in access control revisited. In *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, SACMAT '16, page 101–111, New York, NY, USA, 2016. Association for Computing Machinery.

[CGW17] Jason Crampton, Gregory Z. Gutin, and Rémi Watrigant. On the satisfiability of workflows with release points. In *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies, SACMAT 2017, Indianapolis, IN, USA, June 21-23, 2017*, pages 207–217, 2017.

[Cha09] DavidW. Chadwick. Federated identity management. In Alessandro Aldini, Gilles Barthe, and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design V*, volume 5705 of *Lecture Notes in Computer Science*, pages 96–120. Springer Berlin Heidelberg, 2009.

[CHK14] Jason Crampton, Michael Huth, and JimHuan-Pu Kuo. Authorized workflow schemas: deciding realizability through ltl(f) model checking. *STTT*, 16(1):31–48, 2014.

[CK08] Jason Crampton and Hemanth Khambhammettu. Delegation and satisfiability in workflow systems. In *SACMAT*, pages 31–40, New York, NY, USA, 2008. ACM.

[CKKT11] Tyrone Cadenhead, Vaibhav Khadilkar, Murat Kantarcioglu, and Bhavani Thuraisingham. A language for provenance access control. In *Proceedings of Conference on Data and Application Security and Privacy*, pages 133–144. ACM, 2011.

[CM04] F. Cuppens and A. Miège. Adorbac: an administration model for or-bac. *Comput. Syst. Sci. Eng.*, 19, 2004.

[Com86] Hubert Comon. Sufficient completeness, term rewriting systems and "anti-unification". In Jörg H. Siekmann, editor, *8th International Conference on Automated Deduction*, pages 128–140, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[CON06] DavidW Chadwick, Sassa Otenko, and TuanAnh Nguyen. Adding support to xacml for dynamic delegation of authority in multiple domains. In Herbert Leitold and EvangelosP. Markatos, editors, *Communications and Multimedia Security*, volume 4237 of *Lecture Notes in Computer Science*, pages 67–86. Springer Berlin Heidelberg, 2006.

[CPU+10] É. Contejean, A. Paskevich, X. Urbain, P. Courtieu, O. Pons, and J. Forest. A3pat, an approach for certified automated termination proofs. In *Proc. of the 2010 ACM SIGPLAN workshop on Partial evaluation and program manipulation*, PEPM '10, pages 63–72, New York, NY, USA, 2010. ACM.

[Cra05] J. Crampton. A reference monitor for workflow systems with constrained task execution. In *10th ACM SACMAT*, pages 38–47. ACM, 2005.

[CS14] Jason Crampton and James Sellwood. Path Conditions and Principal Matching: A New Approach to Access Control. In *Proceedings of Symposium on Access Control Models and Technologies*, pages 187–198. ACM, 2014.

[D.03] Box D. Web services policy framework (WS-policy). volume version 1.1. http://www.oasis-open.org/specs/index.php#ws-secpol, 2003.

[DCdVS96] Sabrina De Capitani di Vimercati and Pierangela Samarati. An authorization model for federated systems. In Elisa Bertino, Helmut Kurth, Giancarlo Martella, and Emilio Montolivo, editors, *Computer Security — ESORICS 96*, pages 99–117, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[DDCdVPS02] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. A fine-grained access control system for xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(2):169–202, May 2002.

[DDLS01] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In Morris Sloman, Emil C. Lupu, and Jorge Lobo, editors, *Policies for Distributed Systems and Networks*, pages 18–38, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[DDO08] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in {BPMN}. *Inf. and Soft. Tech.*, 50(12):1281 – 1294, 2008.

[DDS05] E. Damiani, S. De Capitani di Vimercati, and P. Samarati. New paradigms for access control in open environments. In *Proc. of the 5th IEEE International Symposium on Signal Processing and Information*, Athens, Greece, December 2005.

[Der87] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1):69 – 115, 1987.

[DeT02] John DeTreville. Binder, a logic-based security language. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, page 105, USA, 2002. IEEE Computer Society.

[DFJS07] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, and P. Samarati. Access control policies and languages. *International Journal of Computational Science and Engineering (IJCSE)*, 3(2):94–102, 2007.

[DFK06] D. J. Dougherty, K. Fisler, and S. Krishnamurthi. Specifying and reasoning about dynamic access-control policies. In Ulrich Furbach and Natarajan Shankar, editors, *Proc. of IJCAR'06*, volume 4130 of *Lecture Notes in Computer Science*, pages 632–646. Springer, 2006.

[dHZ16a] Jerry den Hartog and Nicola Zannone. A policy framework for data fusion and derived data control. In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*, ABAC '16, page 47–57, New York, NY, USA, 2016. Association for Computing Machinery.

[dHZ16b] Jerry den Hartog and Nicola Zannone. A policy framework for data fusion and derived data control. In *Proceedings of International Workshop on Attribute Based Access Control*, pages 47–57. ACM, 2016.

[DJH12] J. Dubrovin, T.A. Junttila, and K. Heljanko. Exploiting step semantics for efficient bounded model checking of asynchronous systems. *Sci. Comput. Program.*, 77(10-11):1095–1121, 2012.

[DKKdO07] D. J. Dougherty, C. Kirchner, H. Kirchner, and A. Santana de Oliveira. Modular access control via strategic rewriting. In *Proceedings of 12th European Symposium On Research In Computer Security, ESORICS*, pages 578–593, 2007.

[dOWKK07] A. Santana de Oliveira, E. Ke Wang, C. Kirchner, and H. Kirchner. Weaving rewrite-based access control policies. In *Proceedings of the 2007 ACM workshop on Formal methods in security engineering, FMSE 2007, Fairfax, VA, USA, November 2, 2007*, pages 71–80. ACM, 2007.

[dS17] Daniel Ricardo dos Santos. *Automatic Techniques for the Synthesis and Assisted Deployment of Security Policies in Workflow-based Applications.* PhD thesis, University of Trento, Italy, 2017.

[dSMS+16] Daniel Ricardo dos Santos, Roberto Marinho, Gustavo Roecker Schmitt, Carla Merkle Westphall, and Carlos Becker Westphall. A framework and risk assessment approaches for risk-based access control in the cloud. *J. Netw. Comput. Appl.*, 74(C):86–97, 2016.

[dSR17] Daniel Ricardo dos Santos and Silvio Ranise. A survey on workflow satisfiability, resiliency, and related problems. *CoRR*, abs/1706.07205, 2017.

[dSRCP15] Daniel R. dos Santos, Silvio Ranise, Luca Compagna, and Serena E. Ponta. Assisting the deployment of security-sensitive workflows by finding execution scenarios. In Pierangela Samarati, editor, *Data and Applications Security and Privacy XXIX*, pages 85–100. Springer International Publishing, 2015.

[dVS97] S. De Capitani di Vimercati and P. Samarati. Authorization specification and enforcement in federated database systems. *J. Comput. Secur.*, 5:155–188, March 1997.

[EAC98] Guy Edjlali, Anurag Acharya, and Vipin Chaudhary. History-based access control for mobile code. In *Proceedings of Conference on Computer and Communications Security*, pages 38–48. ACM, 1998.

[EBA+07] C. Emig, F. Brandt, S. Abeck, J. Biermann, and H. Klarl. An access control metamodel for web service-oriented architecture. In *Software Engineering Advances, 2007. ICSEA 2007. International Conference on*, pages 57–57. IEEE Computer Society, 2007.

[EMM06] S. Escobar, C. Meadows, and J. Meseguer. A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. *Theor. Comput. Sci.*, 367:162–202, November 2006.

[EP05] R. Echahed and F. Prost. Security policy in a declarative style. In *Proceedings of the 7th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, PPDP '05, page 153–163, New York, NY, USA, 2005. Association for Computing Machinery.

[FA08] David Ferraiolo and Vijay Atluri. A meta model for access control: Why is it needed and is it even possible to achieve? In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, page 153–154, New York, NY, USA, 2008. Association for Computing Machinery.

[FAG11] David Ferraiolo, Vijayalakshmi Atluri, and Serban Gavrila. The policy machine: A novel architecture and framework for access control policy specification and enforcement. *J. Syst. Archit.*, 57(4):412–424, 2011.

[FCKV10] Viktoria Felmetsger, Ludovico Cavedon, Christopher Kruegel, and Giovanni Vigna. Toward automated detection of logic vulnerabilities in web applications. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, page 10, USA, 2010. USENIX Association.

[FGHK05] David F. Ferraiolo, Serban Gavrila, Vincent Hu, and D. Richard Kuhn. Composing and combining policies under the policy machine. In *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*, SACMAT '05, page 11–20, New York, NY, USA, 2005. Association for Computing Machinery.

[FJ95] M. Fernández and J.-P. Jouannaud. Modular termination of term rewriting systems revisited. In *Recent Trends in Data Type Specification. Proc. 10th. Workshop on Specification of Abstract Data Types (ADT'94)*, number 906 in Lecture Notes in Computer Science, Santa Margherita, Italy, 1995.

[FKMT05] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change-impact analysis of access-control policies. In *27th International Conference on Software Engineering (ICSE 2005), 15-21 May 2005, St. Louis, Missouri, USA*, pages 196–205, 2005.

[FMP13] Anna Lisa Ferrara, P Madhusudan, and Gennaro Parlato. Policy analysis for self-administrated role-based access control. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 432–447. Springer Berlin Heidelberg, 2013.

[FSG+01] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[Gab04] Alban Gabillon. An authorization model for xml databases. In *Proceedings of the 2004 Workshop on Secure Web Service*, SWS '04, pages 16–28, New York, NY, USA, 2004. ACM.

[Gar97] V. K. Garg. Methods for Observing Global Properties in Distributed Systems. *IEEE Parallel Distrib. Technol.*, 5(4):69–77, October 1997.

[Gar13] V.K. Garg. Maximal antichain lattice algorithms for distributed computations. In *Proc. of ICDCN*, 2013.

[GB02] Alban Gabillon and Emmanuel Bruno. *Regulating Access to XML Documents*, pages 299–314. Springer US, Boston, MA, 2002.

[GdHN+08] Patrice Godefroid, Peli de Halleux, Aditya Nori, Sriram Rajamani, Wolfram Schulte, Nikolai Tillmann, and Michael Y. Levin. Automated software testing using program analysis. *IEEE Software, Special Issue on Software Development Tools*, 2008.

[Gen98] Thomas Genet. *Contraintes d'ordre et automates d'arbres pour les preuves de terminaison.* Theses, Université Henri Poincaré - Nancy 1, 1998.

[GHHF05] H. Gomi, M. Hatakeyama, S. Hosono, and S. Fujita. A delegation framework for federated identity management. In *ACM Workshop on Digital Identity Management*, pages 94–103. ACM Press, New York, USA, 2005.

[GM15] M. Gario and A. Micheli. pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms. In *SMT Workshop*, 2015.

[GNS17] Ekaterina A. Gorshkova, Boris Novikov, and Manoj Kumar Shukla. A fine-grained access control model and implementation. *Proceedings of the 18th International Conference on Computer Systems and Technologies*, 2017.

[GQ96] L. Gong and X. Qian. Computational Issues in Secure Interoperation. *IEEE Transactions on Software Engineering*, 22(1):43–52, 1996.

[GR10a] S. Ghilardi and S. Ranise. Backward reachability of array-based systems by SMT solving: Termination and invariant synthesis. *In LMCS, Vol. 6, Issue 4*, 2010.

[GR10b] S. Ghilardi and S. Ranise. MCMT: A Model Checker Modulo Theories. In *IJCAR*, volume 6173 of *LNCS*, pages 22–29, 2010.

[GSKT06] J. Giesl, P. Schneider-Kamp, and R. Thiemann. Aprove 1.2: Automatic termination proofs in the dependency pair framework. In *Proceedings IJCAR '06*, LNAI 4130, pages 281–286. Springer, 2006.

[HAM15] Julius Holderer, Rafael Accorsi, and Günter Müller. When four-eyes become too much: A survey on the interplay of authorization constraints and workflow resilience. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, page 1245–1248, New York, NY, USA, 2015. Association for Computing Machinery.

[HCM05] J. Hendrix, M. Clavel, and J. Meseguer. A sufficient completeness reasoning tool for partial specifications. In *Proc. of 16th Int. Conference on Term Rewriting and Applications, RTA 2005, Nara, Japan, 2005*, number 3467 in Lecture Notes in Computer Science, pages 165–174. Springer, 2005.

[HFK+15] V. Hu, D. F. Ferraiolo, D. Kuhn, R. N. Kacker, and Y. Lei. Implementing and managing policy rules in attribute based access control. In *2015 IEEE International Conference on Information Reuse and Integration (IRI)*, volume 1, pages 518–525, Los Alamitos, CA, USA, 2015. IEEE Computer Society.

[HKF15] V. C. Hu, D. Kuhn, and D. F. Ferraiolo. Attribute-based access control. *Computer*, 48(02):85–88, 2015.

[HM05] Nao Hirokawa and Aart Middeldorp. Tyrolean termination tool. In Jürgen Giesl, editor, *Term Rewriting and Applications*, pages 175–184, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc., USA, 1985.

[HRU76a] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, 1976.

[HRU76b] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, 1976.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980.

[Hug05] et al. Hughes, J. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS, March 2005. Available at `http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf`.

[HW08] Joseph Y. Halpern and Vicky Weissman. Using first-order logic to reason about policies. *ACM Trans. Inf. Syst. Secur.*, 11(4), 2008.

[JBLG05] James B. D. Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. on Knowl. and Data Eng.*, 17(1):4–23, 2005.

[JD94] D. Jonscher and K. R. Dittrich. An approach for building secure database federations. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 24–35, San Francisco, CA, USA, 1994.

[Jim01] Trevor Jim. Sd3: A trust management system with certified evaluation. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, page 106, USA, 2001. IEEE Computer Society.

[JKS12a] X. Jin, R. Krishnan, and R. Sandhu. A unified attribute-based access control model covering dac, mac and rbac. In N. Cuppens-Boulahia, F. Cuppens, and

J. Garcia-Alfaro, editors, *Data and Applications Security and Privacy XXVI*, pages 41–55, Berlin, Heidelberg, 2012. Springer.

[JKS12b] Xin Jin, Ram Krishnan, and Ravi Sandhu. A unified attribute-based access control model covering dac, mac and rbac. In *Proceedings of the 26th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy*, DBSec'12, pages 41–55, Berlin, Heidelberg, 2012. Springer-Verlag.

[JLT+08] Somesh Jha, Ninghui Li, Mahesh Tripunitara, Qihua Wang, and William Winsborough. Towards formal verification of role-based access control policies. *IEEE Transactions on Dependable and Secure Computing*, 5(4):242–255, 2008.

[JM06] Mathieu Jaume and Charles Morisset. A formal approach to implement access control. *Journal of Information Assurance and Security*, 1(2):137–148, 2006.

[Jor87] C.S. Jordan. Guide to understanding discretionary access control in trusted systems, 1987.

[JS05] R. Jagadeesan and V. Saraswat. Timed Constraint Programming: A Declarative Approach to Usage Control. In *Proc. 7th ACM-SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP'05)*. ACM Press, 2005.

[JSSS01] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, June 2001.

[JT01] Trent Jaeger and Jonathon E. Tidswell. Practical safety in flexible access control models. *ACM Trans. Inf. Syst. Secur.*, 4(2):158–190, May 2001.

[KAAI21] Nadine Kashmar, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. A review of access control metamodels. *Procedia Computer Science*, 184:445–452, 2021. The 12th International Conference on Ambient Systems, Networks and Technologies (ANT).

[KAI21] Nadine Kashmar, Mehdi Adda, and Hussein Ibrahim. HEAD metamodel: Hierarchical, extensible, advanced, and dynamic access control metamodel for dynamic and heterogeneous structures. *Sensors*, 21(19):6507, 2021.

[Kar15] Günter Karjoth. Aligning security and business objectives for process-aware information systems. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15, page 243, New York, NY, USA, 2015. Association for Computing Machinery.

[KB83] D. E. Knuth and P. B. Bendix. *Simple Word Problems in Universal Algebras*, pages 342–376. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.

[KB07] David Kyle and José Carlos Brustoloni. Uclinux: A linux security module for trusted-computing-based usage controls enforcement. In *Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing*, STC '07, pages 63–70, New York, NY, USA, 2007. ACM.

[KBB+03] AAE. Kalam, R.E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A Miege, C. Saurel, and G. Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 120–131, 2003.

[KF12] Arif Akram Khan and Philip W. L. Fong. Satisfiability and feasibility in a relationship-based workflow authorization model. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security – ESORICS 2012*, pages 109–126, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[KKdO09] Claude Kirchner, Hélène Kirchner, and Anderson Santana de Oliveira. Analysis of rewrite-based access control policies. *Electr. Notes Theor. Comput. Sci.*, 234:55–75, 2009.

[KKP+15] Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-c: A software analysis perspective. *Formal Aspects of Computing*, 27(3):573–609, 2015.

[KL10] A.H. Karp and Jun Li. Solving the transitive access problem for the services oriented architecture. In *Availability, Reliability, and Security, 2010. ARES '10 International Conference on*, pages 46–53, Feb 2010.

[KM16] Kyriakos Kritikos and Philippe Massonet. An integrated meta-model for cloud application security modelling. *Procedia Computer Science*, 97:84–93, 2016.

[KMPP01] M. Koch, L. V. Mancini, and F. Parisi-Presicce. On the specification and evolution of access control policies. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, SACMAT '01, page 121–130, New York, NY, USA, 2001. Association for Computing Machinery.

[KMPP04] M. Koch, L. Mancini, and F. Parisi-Presicce. A graph based formalism for RBAC. In *Proc. of SACMAT 2004, 9th ACM Symposium on Access Control Models and Technologies, New York, USA, 2004*, pages 129–187, 2004.

[KNRZ91] Deepak Kapur, Paliath Narendran, Daniel J. Rosenkrantz, and Hantao Zhang. Sufficient-completeness, ground-reducibility and their complexity. *Acta Informatica*, 28(4):311–350, 1991.

[KNS08] Karl Krukow, Mogens Nielsen, and Vladimiro Sassone. A logical framework for history-based access control and reputation systems. *J. Comput. Secur.*, 16(1):63–101, 2008.

[Lam71] Butler W. Lampson. Protection. In *5th Princeton Conference on Information Sciences and Systems*, page 437, 1971.

[Lan66] P. J. Landin. The next 700 programming languages. *Commun. ACM*, 9(3):157–166, 1966.

[LB13] Julien Lacroix and Omar Boucelma. Provenance-based access control in the cloud. In *Proceedings of International Conference on Services Computing*, pages 755–756. IEEE, 2013.

[LBOG06] Adam J. Lee, Jodie P. Boyer, Lars E. Olson, and Carl A. Gunter. Defeasible security policy composition for web services. In *Proceedings of the Fourth ACM Workshop on Formal Methods in Security*, FMSE '06, page 45–54. Association for Computing Machinery, 2006.

[Liu07] Alex X. Liu. Change-impact analysis of firewall policies. In Joachim Biskup and Javier López, editors, *Computer Security – ESORICS 2007*, pages 155–170, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[LK07] Jun Li and Alan H. Karp. Access control for the services oriented architecture. In *Proceedings of the 2007 ACM Workshop on Secure Web Services*, SWS '07, pages 9–17. ACM, 2007.

[Llo84] J. W. Lloyd. *Foundations of Logic Programming*. Springer, 1984.

[LM03] N. Li and J. C. Mitchell. Datalog with constraints: a foundation for trust management languages. In *PADL'03*, pages 58–73, 2003.

[LMRM12] Maria Leitner, Juergen Mangler, and Stefanie Rinderle-Ma. Definition and enactment of instance-spanning process constraints. In X. Sean Wang, Isabel Cruz, Alex Delis, and Guangyan Huang, editors, *Web Information Systems Engineering - WISE 2012*, pages 652–658, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[LMW02] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, page 114, USA, 2002. IEEE Computer Society.

[LRM14] M. Leitner and S. Rinderle-Ma. A systematic review on security in process-aware information systems–constitution, challenges, and future directions. *Inf. and Soft. Tech.*, 56(3):273–293, 2014.

[LSWY07] Adam J. Lee, Kent E. Seamons, Marianne Winslett, and Ting Yu. *Automated Trust Negotiation in Open Systems*, pages 217–258. Springer US, Boston, MA, 2007.

[LT06] Ninghui Li and Mahesh V Tripunitara. Security analysis in role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 9(4):391–420, 2006.

[LTL10] Y. Long, Z. Tang, and X. Liu. Attribute mapping for cross-domain access control. In *2010 International Conference on Computer and Information Application*, volume 1, pages 343–347, 2010.

[LWQ+09] N. Li, Q. Wang, W. H. Qardaji, E. Bertino, P. Rao, J. Lobo, and D. Lin. Access control policy combining: theory meets practice. In *SACMAT 2009, 14th ACM Symposium on Access Control Models and Technologies, Stresa, Italy, June 3-5, 2009, Proceedings*, pages 135–144. ACM, 2009.

[MA04] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[Mao10] Yun Mao. On the declarativity of declarative networking. *SIGOPS Oper. Syst. Rev.*, 43(4):19–24, January 2010.

[Mas97] Fabio Massacci. Reasoning about security: A logic and a decision method for role-based access control. In Dov M. Gabbay, Rudolf Kruse, Andreas Nonnengart, and Hans Jürgen Ohlbach, editors, *Qualitative and Quantitative Practical Reasoning*, pages 421–435, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[McA04] N. McAllister. Toward a federated future. *InfoWorld*, 26(36):44–48, 2004.

[MCF+11] Luc Moreau, Ben Cliffor, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale,

Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756, 2011.

[McG09] R. McGraw. Risk-adaptable access control (radac). In *Proc. of NIST Privilege (Access) Management Workshop*, 2009.

[Miè05] Alexandre Miège. *Definition of a formal framework for specifying security policies. The Or-BAC model and extensions.* Theses, Télécom ParisTech, 2005.

[MKD19] M. Marin, T. Kutsia, and B. Dundua. A rule-based approach to the decidability of safety of abacα. In *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies, SACMAT 2019, Toronto, ON, Canada, June 03-06, 2019*, pages 173–178, New York, 2019. ACM Press.

[MKT05] P. Madsen, Y. Koga, and K. Takahashi. Federated identity management for protecting users from id theft. In *ACM Workshop on Digital Identity Management*, pages 77–83. ACM Press, New York, USA, 2005.

[MM13] Luc Moreau and Paolo Missier. PROV-DM: The PROV Data Model. W3C Recommendation, W3C, 2013.

[MMM14] J.C. Mace, C. Morisset, and A. Moorsel. Quantitative workflow resiliency. In *Proc. of ESORICS*, 2014.

[MOPB06] Massimo Mecella, Mourad Ouzzani, Federica Paci, and Elisa Bertino. Access control enforcement for conversation-based web services. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 257–266, New York, NY, USA, 2006. ACM.

[Mor07] Charles Morisset. *Sémantique des systèmes de contrôle d'accès : définition d'un cadre sémantique pour la spécification, l'implantation et la comparaison de modèles de contrôle d'accès.* PhD thesis, Thèse de doctorat Informatique, Université Paris 6, 2007.

[MP04] Deborah L. McGuinness and Paulo Pinheiro da Silva. Explaining answers from the semantic web: the inference web approach. *Journal of Web Semantics*, 1(4):397 – 413, 2004. International Semantic Web Conference 2003.

[MS18] Charles Morisset and David Sanchez. Visabac: A tool for visualising ABAC policies. In Paolo Mori, Steven Furnell, and Olivier Camp, editors, *Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Madeira - Portugal, January 22-24, 2018.*, pages 117–126. SciTePress, 2018.

[MT07] José Meseguer and Prasanna Thati. Symbolic Reachability Analysis Using Narrowing and Its Application to Verification of Cryptographic Protocols. *Higher Order Symbol. Comput.*, 20(1-2):123–160, June 2007.

[MU04] Claude Marché and Xavier Urbain. Modular and incremental proofs of acctermination. *Journal of Symbolic Computation*, 38(1):873 – 897, 2004.

[Mur89] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[New42] M.H.A. Newman. On theories with a combinatorial definition of equivalence. *Annals of Mathematics*, 43(2):223–243, 1942.

[NPS13] Dang Nguyen, Jaehong Park, and Ravi S. Sandhu. A provenance-based access control model for dynamic separation of duties. In *Proceedings of Annual International Conference on Privacy, Security and Trust*. IEEE, 2013.

[NR09] Juan A. Navarro and Andrey Rybalchenko. Operational semantics for declarative networking. In Andy Gill and Terrance Swift, editors, *Practical Aspects of Declarative Languages*, pages 76–90, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[NRZ+15] Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff. How amazon web services uses formal methods. *Commun. ACM*, 58(4):66–73, March 2015.

[OAS03] OASIS. eXtensible Access Control Markup language (XACML), 2003. http://www.oasis-open.org/xacml/docs/.

[OAS13] OASIS. eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard, 2013.

[oD85] Department of Defense. Trusted computer system evaluation criteria. 5200(28), 1985.

[OL10] Alan C O'Connor and Ross J Loomis. Economic analysis of role-based access control, 2010.

[PG03] M. P. Papazoglou and D. Georgakopulos. Service Oriented Computing. *Comm. of the ACM*, 46(10):25–38, 2003.

[Pha10] Q. Pham. *Delegation Framework for Federated Systems*. PhD thesis, Information Security Institute, Queensland University of Technology, Brisbane, Australia, 2010.

[PHB+08] A. Pretschner, M. Hilty, D. Basin, C. Schaefer, and T. Walter. Mechanisms for usage control. In *Proceeding ASIACCS '08 Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 240–244. ACM, New York, 2008.

[PMH07] Alexander Pretschner, Fabio Massacci, and Manuel Hilty. Usage control in service-oriented architectures. In Costas Lambrinoudakis, Günther Pernul, and A. Min Tjoa, editors, *Trust, Privacy and Security in Digital Business*, pages 83–93, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[PNS12] Jaehong Park, Dang Nguyen, and Ravi Sandhu. A provenance-based access control model. In *Proceedings of the 2012 Tenth Annual International Conference on Privacy, Security and Trust (PST)*, PST '12, pages 137–144, Washington, DC, USA, 2012. IEEE Computer Society.

[Pre47] Associated Press. Bill providing highway access control drafted, 1947.

[PS02] Jaehong Park and Ravi Sandhu. Towards usage control models: Beyond traditional access control. In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*, SACMAT '02, pages 57–64, New York, NY, USA, 2002. ACM.

[PS04] Jaehong Park and Ravi Sandhu. The ucon-abc usage control model. *ACM Transactions in Information Systems Security*, 7(1):128–174, 2004.

[PSZ18] Federica Paci, Anna Squicciarini, and Nicola Zannone. Survey on access control for community-centered collaborative systems. *ACM Comput. Surv.*, 51(1):6:1–6:38, 2018.

[PY14] J. Pei and X. Ye. Towards Policy Retrieval for Provenance Based Access Control Model. In *Proceedings of International Conference on Trust, Security and Privacy in Computing and Communications*, pages 769–776. IEEE, 2014.

[PYL13] I. Ray P. Yang, X. Xie and S. Lu. Satisfiability analysis of workflows with control-flow patterns and authorization constraints. *IEEE TSC*, 99, 2013.

[RFR+12] Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, page 323–334, New York, NY, USA, 2012. Association for Computing Machinery.

[RSM+15] A. Roy, S. Sural, A.K. Majumdar, J. Vaidya, and V. Atluri. Minimizing organizational user requirement while meeting security constraints. *ACM Trans. Manage. Inf. Syst.*, 6(3):12:1–12:25, 2015.

[San08] A. Santana de Oliveira. *Réécriture et Modularité pour les Politiques de Sécurité*. PhD thesis, Université Henri Poincaré, Nancy, France, 2008.

[SAS04] D. Shin, G. J. Ahn, and P. Shenoy. Ensuring information assurance in federated identity management. In *IEEE Int. Conf. on Performance, Computing, and*

Communications, pages 821–826. IEEE Computer Society, Washington DC, USA, 2004.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, February 1996.

[Sch00] Fred B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.

[SdV01] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design*, pages 137–196, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Sha93] A. U. Shankar. An Introduction to Assertional Reasoning for Concurrent Systems. *ACM Comput. Surv.*, 25(3):225–262, September 1993.

[SIM+07] M. Srivatsa, A. Iyengar, T. Mikalsen, I. Rouvellou, and Jian Yin. An access control system for web service compositions. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 1–8, July 2007.

[SKAL12] Bernard Stepien, Hemanth Khambhammettu, Kamel Adi, and Luigi Logrippo. Catbac: A generic framework for designing and validating hybrid access control models. *2012 IEEE International Conference on Communications (ICC)*, pages 6721–6726, 2012.

[spe03] OASIS specifications. Security assertion markup language (saml), 2003.

[SPNS16] L. Sun, J. Park, D. Nguyen, and R. Sandhu. A provenance-aware access control framework with typed provenance. *IEEE Transactions on Dependable and Secure Computing*, 13(4):411–423, 2016.

[SS04] Christian Skalka and Scott Smith. History effects and verification. In *Programming Languages and Systems*, pages 107–128. Springer, 2004.

[SSM03] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Petri net analysis using invariant generation. In *In Verification: Theory and Practice, LNCS 2772*, pages 682–701. Springer Verlag, 2003.

[ST98] C. C. Shilakes and J. Tylman. *Enterprise Information Portals.* Merrill Lynch, Inc., New York, NY, 1998.

[ST15] R. Sebastiani and P. Trentin. OptiMathSAT: A Tool for Optimization Modulo Theories. In *Proc. of CAV*, 2015.

[SYRG07] Scott D Stoller, Ping Yang, C R Ramakrishnan, and Mikhail I Gofman. Efficient policy analysis for administrative role based access control. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 445–455. ACM, 2007.

[SYSR06] Amit Sasturkar, Ping Yang, Scott D Stoller, and CR Ramakrishnan. Policy analysis for administrative role based access control. In *Computer Security Foundations Workshop, 2006. 19th IEEE*, pages 13–pp. IEEE, 2006.

[SYTB13] Wei She, I-Ling Yen, B. Thuraisingham, and E. Bertino. Security-aware service composition with fine-grained information flow control. *Services Computing, IEEE Transactions on*, 6(3):330–343, July 2013.

[Ter03] Terese. *Term Rewriting Systems.* Cambridge University Press, 2003.

[TFS18] Lakshya Tandon, Philip W. L. Fong, and Reihaneh Safavi-Naini. HCAP: A History-Based Capability System for IoT Devices. In *Proceedings of Symposium on Access Control Models and Technologies*, pages 247–258. ACM, 2018.

[TK06] Michael Carl Tschantz and Shriram Krishnamurthi. Towards reasonability properties for access-control policy languages. In *Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies*, SACMAT '06, page 160–169, New York, NY, USA, 2006. Association for Computing Machinery.

[TS97] Roshan K. Thomas and Ravi S. Sandhu. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented autorization management. In *Database Securty XI: Status and Prospects, IFIP TC11 WG11.3 Eleventh International Conference on Database Security, 10-13 August 1997, Lake Tahoe, California, USA*, volume 113 of *IFIP Conference Proceedings*, pages 166–181, 1997.

[TWM+09] Mohit Tiwari, Hassan M.G. Wassel, Bita Mazloom, Shashidhar Mysore, Frederic T. Chong, and Timothy Sherwood. Complete information flow tracking from the gates up. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, page 109–120, New York, NY, USA, 2009. Association for Computing Machinery.

[TZG+13] Daniel Trivellato, Nicola Zannone, Maurice Glaundrup, Jacek Skowronek, and Sandro Etalle. A semantic security framework for systems of systems. *Int. J. Cooperative Inf. Syst.*, 22(1), 2013.

[UBR14] Worachet Uttha, Clara Bertolissi, and Silvio Ranise. Towards a reference architecture for access control in distributed web applications. In Wouter Joosen, Fabio Martinelli, and Thomas Heyman, editors, *Proceedings of the 2014 ESSoS Doctoral Symposium co-located with the International Symposium on Engineering Secure Software and Systems (ESSoS 2014), Munich, Germany, February 26, 2014*, volume 1298 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[UBR15] W. Uttha, C. Bertolissi, and S. Ranise. Modeling Authorization Policies for Web Services in Presence of Transitive Dependencies and their Enforcement in an Extended IAM Architecture. In *SECRYPT 2015 - Proceedings of the 12th International Conference on Security and Cryptography, Colmar, Alsace, France, 20-22 July, 2015*, pages 293–300. SciTePress, 2015.

[Utt16] Worachet Uttha. *Etude des politiques de sécurité pour les applications distribuées : le problème des dépendances transitives : modélisation, vérification*

et mise en oeuvre. PhD thesis, Thèse de doctorat Informatique Aix-Marseille Université 2016, 2016.

[vdAH03] W.M.P. van der Aalst and A. H. M. Ter Hofstede. Yawl: Yet another workflow language. *Inf. Systems*, 30:245–275, 2003.

[vdAtHKB03] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[vdAvHtH+11] W.M.P. van der Aalst, K.M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, and M.T. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Comp.*, 23(3):333–363, 2011.

[Vig05] L. Viganò. Automated security protocol analysis with the AVISPA tool. In *Proc. of MFPS'05*, volume 155 of *ENTCS*, pages 61–86. Elsevier, 2005.

[WA06] Janice Warner and Vijayalakshmi Atluri. Inter-instance authorization constraints for secure workflow management. In *Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies*, SACMAT '06, page 190–199, New York, NY, USA, 2006. Association for Computing Machinery.

[WFM96] WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.

[WJ03a] D. Wijesekera and S. Jajodia. A propositional policy algebra for access control. *ACM Trans. Inf. Syst. Secur.*, 6(2):286–325, 2003.

[WJ03b] Duminda Wijesekera and Sushil Jajodia. A propositional policy algebra for access control. *ACM Trans. Inf. Syst. Secur.*, 6(2):286–325, 2003.

[WKB07] Jacques Wainer, Akhil Kumar, and Paulo Barthelmess. Dw-rbac: A formal security model of delegation and revocation in workflow systems. *Information Systems*, 32(3):365–384, 2007.

[WL10] Q. Wang and N. Li. Satisfiability and resiliency in workflow authorization systems. *TISSeC*, 13:40:1–40:35, December 2010.

[WW06] Z. Wu and A. C. Weaver. Requirements of federated trust management for service-oriented architectures. In *4th Int. Conf. on Privacy, Security, and Trust*, Ontario, Canada, 2006.

[YD09] A. Abou El Kalam Y. Deswarte. *Poly-OrBAC: An access control model for inter-organizational web services.* IGI-Global, 2009.

[Yin07] Mudhakar Srivatsa ; Arun Iyengar ; Thomas Mikalsen ; Isabelle Rouvellou ; Jian Yin. An access control system for web service compositions. In *IEEE International Conference on Web Services (ICWS 2007)*, 2007.

[YT02] M. I. Yagüe and J. M. Troya. A semantic approach for access control in web services. In *Proceedings of the 2002 International Conference on EuroWeb*, EuroWeb'02, pages 3–3, Swindon, UK, 2002. BCS Learning & Development Ltd.

[YT05] Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *Proceedings of the IEEE International Conference on Web Computer Society.*

[Zan95] *H. Zantema. Termination of term rewriting by semantic labelling.* Fundam. Inf.*, 24(1–2):89–105, 1995.*

[ZHLL05] *Chen Zhao, Nuermaimaiti Heilili, Shengping Liu, and Zuoquan Lin. Representation and reasoning on rbac: A description logic approach. In Dang Van Hung and Martin Wirsing, editors,* Theoretical Aspects of Computing – ICTAC 2005*, pages 381–393, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.*

[ZJW06] *Nicola Zannone, Sushil Jajodia, and Duminda Wijesekera. Creating Objects in the Flexible Authorization Framework. In* Data and Applications Security XX*, pages 1–14. Springer, 2006.*

[ZOS03] *Xinwen Zhang, Sejong Oh, and Ravi Sandhu. Pbdm: A flexible delegation model in rbac. In* Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, SACMAT '03, pages 149–157, New York, NY, USA, 2003. ACM.*

[ZZ08] *Mingsheng Zhang and Mingyi Zhang. An approach for handling conflicts in authorization.* Wuhan University Journal of Natural Sciences*, 13(5):626, 2008.*