Hypergraph-based Access Control using Formal Language Expressions – $\mathcal{HGAC}$

Alexander Lawall

Institute for Information Systems at Hof University, Alfons-Goppel-Platz 1, 95028 Hof, Germany

Keywords: Access Control, Attribute-based Access Control, Language Expressions, Organizational Model, Identity Management.

Abstract: In all organizations, access assignments are essential in order to ensure data privacy, permission levels and the correct assignment of tasks. Traditionally, such assignments are based on total enumeration, with the consequence that constant effort has to be put into maintaining the assignments. This problem still persists when using abstraction layers, such as group and role concepts, e.g. Access Control Matrix and Role-Based Access Control. Role and group memberships are statically defined and members have to be added and removed constantly. This paper describes a novel approach – Hypergraph-Based Access Control $\mathcal{HGAC}$ – to assign human and automatic subjects to access rights in a declarative manner. The approach is based on an organizational (meta-) model and a declarative language. The language is used to express queries and formulate predicates. Queries define sets of subjects based on their properties and their position in the organizational model. They also contain additional information that causes organizational relations to be active or inactive depending on predicates. In $\mathcal{HGAC}$, the subjects that have a specific permission are determined by such a query. The query itself is not defined statically but created by traversing a hypergraph path. This allows a structured aggregation of permissions on resources. Consequently, multiple resources can share parts of their queries.

1 INTRODUCTION

Nowadays, companies have to deal with permanent change. Subjects (e.g. human actors, machines, printers, etc.) join, leave or move within the organization. The rearrangement of whole departments is also common. Therefore, the flexibility of the organizational structure is essential to react to such changes, cf. (Vahs, 2007). Otherwise, the organization risks to lose their partners, e.g. deliverers and customers, and elimination from the market, cf. (Krcmar, 2010).

The organizational structure is currently shaped by work in project teams, global teams, networks and global teams in networks (cf. (Krcmar, 2010)). (Lawall et al., 2014b) formalizes a metamodel for modeling arbitrary organization structures that provides the required flexibility and complexity.

Access assignments have to be appropriate to policies that are declared in the company, cf. (Ferrari, 2010, p. 4). Consequently, the validity of relations has to be restricted to realize these policies, cf. (Lawall et al., 2014d). This is fulfilled by different types of predicates assigned to the relations.

Language expressions restrict the validity based on context information, parameters handed from application systems and / or attributes of subjects respectively resources in a company (cf. (Lawall et al., 2014a)). Figure 3 shows an example of a restricted relation based on parameters. The restricted relation with the language expression `damage > "1500"` is traversed if the parameter handed from the application system corresponding to “damage” fulfills the predicate.

A hyperedge in the organizational model – also called hyper-relation – restricts a relation by role. This means that a relation in the organizational model is only traversed if an entity acts in the appropriate role, cf. (Lawall et al., 2014d) and (Lawall et al., 2014a). Figure 3 depicts a hyperedge between two subjects and a role. The relation is traversed if the origin subject acts in the role in that the hyperedge ends in.

This contribution compares approaches for defining access assignments and establishes the novel $\mathcal{HGAC}$. The conjunction of three components – organizational model, declarative language and hypergraph-based access control – are a powerful mechanism to declare access assignments that are stable over time and fit the organizational circumstances and policies.

The formulated language expression (query) is used to define the access assignments in application systems. It defines policies of the company. Policies do not change often, and neither does the expression. Consequently, all application systems create no maintenance effort concerning access assignments after the initial definition of the expressions. Every time a subject joins, leaves or moves within the company, only logically centralized organizational model, cf. section 6, has to be changed. The application systems are not affected.

Knorr describes in (Knorr, 2000) an approach to assign access rights by workflows that are modeled with petri nets. If a subject is assigned to a task in a process, the subject automatically gets the rights to objects needed to execute the task. The definition of responsible subjects is done by role-based access control (RBAC). The approach is only valid in workflows. All other application systems are excluded[1].

Outline

In order to demonstrate widespread key issues, sections 2.1 and 2.2 describe the access control matrix and the role-based access control.

Chapter 3 then introduces the novel approach called hypergraph-based access control. After the definition of hypergraphs an example of an organizational model is described (section 3.1). Section 3.2 defines the formal specification of $\mathcal{HGAC}$ which is explained by example in section 3.3.

The subsequent chapter 4 illustrates the proposed declarative language with syntax (section 4.1) and semantics (section 4.2). The language is used to declare the subjects that are assigned to access rights.

The paper concludes with a case study (chapter 5) and the overall conclusion of the contribution (chapter 6).

2 ACCESS CONTROL

For the definition of access rights exist different approaches. The following sections describe the access control matrix (ACM) and the wide-spread role-based access control (RBAC).

The basic model of access control consists generally of a tuple, cf. (Benantar, 2006, pp. 22) and (Ferrari, 2010, p. 6). It consists of sets $\mathcal{S}, \mathcal{R}$ and $\mathcal{O}$. The set $\mathcal{S}$ includes all subjects substantiated by enumeration and represents users and processes. The access rights $\mathcal{R}$, e.g. *read, write* for files and *execute* for processes, are the operations on concrete objects of the set $\mathcal{O}$. The elements of $\mathcal{O}$ - files, processes, tables, devices, and so on - are the objects on which subjects have access rights.

> "There is usually a direct relationship between the cost of administration and the number of associations that must be managed in order to administer an access control policy: The larger the number of associations, the costlier and more error-prone access control administration." (Ferraiolo et al., 2003, p. 19)

A concrete access right $\mathcal{Z}$ is defined as $\mathcal{Z} = (s, r, o)$ with $s \in \mathcal{S}, r \in \mathcal{R}$ and $o \in \mathcal{O}$. In general, there are two variations to define access rights. All subjects have all access rights on all objects except rights that are explicitly revoked with tuples $\mathcal{Z}$. Another concept is that no subject has any access rights on any object until the access right is explicitly defined. The second case is the most used approach[2].

2.1 Access Control Matrix – ACM

The basic idea of the access control matrix was introduced in (Graham and Denning, 1972). The formalization of (Saunders et al., 2001) and (Seufert, 2002) is used to describe the access control matrix (ACM).

The configuration of a concrete access control matrix is defined with $ACM = (\mathcal{S}, \mathcal{R}, \mathcal{O}, (R^t)_{s \in \mathcal{S}, o \in \mathcal{O}})$[3]. The access control matrix $(R^t)_{s \in \mathcal{S}, o \in \mathcal{O}}$ consists of elements $R^t \subseteq \mathcal{R}$, where subjects $s \in \mathcal{S}$ are represented as rows and objects $o \in \mathcal{O}$ are represented as columns. An entry R^t in the matrix is the access right R^t of subject s to object o (see figure 1).

A configuration in an application system (e.g. workflow management systems, internet portals, database management systems, enterprise resource planning systems) with processes and files is given with:

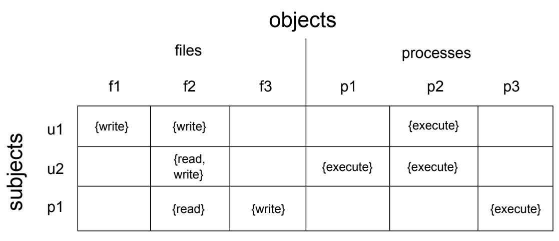
- $\mathcal{S} = \{u1, u2, p1\}$ is the set of users $u1, u2$ and process $p1$
- $\mathcal{R} = \{\text{read, write, execute}\}$ is the set of rights for processes (execute) and files (read, write)
- $\mathcal{O} = \{f1, f2, f3, p1, p2, p3\}$ is the set of objects with files and processes

[1] The approach is only suited for process-oriented organizational structures. The functional and divisional perspective is omitted.

[2] This case is used in the remaining paper.

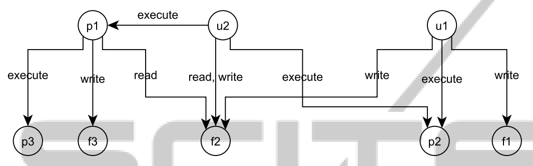
[3] t indicates the point of time of a configuration in a system.

- $(R^t)_{s \in \mathcal{S}, o \in O}$ is represented as figure 1a respectively figure 1b



		objects					
		files			processes		
		f1	f2	f3	p1	p2	p3
subjects	u1	{write}	{write}			{execute}	
	u2		{read, write}		{execute}	{execute}	
	p1		{read}	{write}			{execute}

(a) Configuration of an Access Control Matrix (cf. (Seufert, 2002))



(b) Access Control Matrix as graph

Figure 1: Representations of an Access Control Matrix.

Discussion

The static definition of subjects assigned with access rights on objects is a problem regarding the continuous change in companies. Especially human subjects joining, moving or leaving companies are of interest concerning access rights and policies. Subjects are not limited to persons. Also automatic subjects like machines, computers and agents (cf. (Lawall et al., 2013b)) are involved. Software and hardware are similarly fluctuating in companies. The combination of the different application systems $\mathcal{K}$ and applications $\mathcal{A}$ makes the validity of access rights / policies at a specific point in time error-prone. A consistent state across all application systems is almost impossible.

Another aspect is the high maintenance effort resulting from continuous changes. The administrators, the responsible users or both are challenged by this effort.

The approximated maintenance effort in ACM for subject $s \in \mathcal{S}$ joining $j(s)$, moving $m(s)$ or leaving $l(s)$ is:

$$ACM_{j(s),m(s)} = \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} |O_{s,k,a}| \cdot |\mathcal{R}_{k,a}| \qquad (1)$$

$$ACM_{l(s)} = \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} |O_{s,k,a}| \qquad (2)$$

The determination of all subjects that are assigned to access right r on an object o is another problem. All subjects of a company have to be resolved from the involved column o. Each entry in the access control matrix has to be compared with the right r to decide if the subject is assigned to the right. In the worst case $|\mathcal{S}| \cdot |\mathcal{R}|$ comparisons per object are needed.

2.2 Role-Based Access Control – RBAC

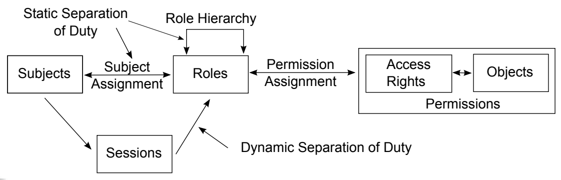


Figure 2: Role-based Access Control adapted by (Ferraiolo et al., 1999).

Instead of assigning subjects individually to objects with their concrete access rights like in section 2.1 – the access control matrix – subjects are associated with one or more roles (*User Assignment, UA*). A role is associated with a corresponding set of access rights to objects (*Permission Assignment, PA*). A subject's access to objects is based on the access rights of the roles to which the subject is assigned, cf. (Ferraiolo et al., 2001), (Liu et al., 2006), (Sandhu, 1992), (Sandhu, 1998) and (Sandhu et al., 1996).
Administrators for identity management tasks have to manage the access rights to an ideally small number of role definitions, rather than many individual user permissions, cf. (Williamson et al., 2009).

There are more RBAC implementations extending the mentioned core RBAC. In (Chen, 2011) and (Chen and Zhang, 2011), the extensions of RBAC include role hierarchies, constraints and the combination of role hierarchies and constraints (cf. fig. 2). Role hierarchies are used to inherit access rights. For example, a head of a department is superior to his clerk and has also same access rights to all objects which the clerk is assigned to.
The constraint extension restricts the *Subject Assignment* and the assignment in role hierarchies with *Static Separation of Duty* in RBAC with constraints. *Dynamic Separation of Duty* restricts the active roles of a subject in a session, cf. figure 2.

Discussion

The main factor for using RBAC compared to the access control matrix is to reduce management costs. If access rights are assigned to a subject's role, the maintenance effort for managing individual access rights is eliminated, cf. (Ferraiolo et al., 2003, p. 19). This means that as a subject moves into or out of a job function within an organization, access to the associated roles is granted and automatically rescinded.

The administration effort is decreased because the reassignment of subjects to roles compared to the assignment of subjects to access rights has less work load. If there are more roles than subjects needed, the effort is higher.

The problem remains that the new role subject assignment has to be done in all application systems $\mathcal{K}$ and applications $\mathcal{A}$. Another aspect is that the *User Assignment* is static with regards to access rights. The access rights in a company for objects (e.g. processes, files,...) are often based on *context-*[4], *attribute-*[5] and / or *parameter-*[6]values. This is difficult with the RBAC approach. For each characteristic policy, a separate role is needed. Technical roles (roles in applications) are no longer job functions as intended in RBAC, cf. (Ferraiolo et al., 2003, p. 10). Thus, the organizational job functions are not congruent to the roles in RBAC. An permanent effort in maintaining the mapping between job functions and roles is essential to ensure consistent policies / access rights. Thus a consistent access right assignment spread over all application systems with fluctuating subjects is hardly possible. The approximated maintenance effort in RBAC for subject $s \in \mathcal{S}$ joining $j(s)$, moving $m(s)$ or leaving $l(s)$ is:

$$RBAC_{j(s),m(s),l(s)} = \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} |Role_{s,k,a}| \qquad (3)$$

The permission definition based on RBAC is more stable over time than using the access control matrix in the application systems respectively applications. Because the assignment of a role to permissions remains the same, the *Subject Assignment* changes.

Using RBAC, the determination of all subjects assigned to access right r on an object o is the following: Subjects are assigned to their roles by *UA* and permissions are assigned to roles by *PA*. In order to determine all assigned subjects, all roles with the right r on o need to be found by evaluating *PA*. In a second step, all subjects assigned to these roles have to be resolved by evaluating *UA*.

3 HYPERGRAPH-BASED ACCESS CONTROL – $\mathcal{HGAC}$

The formal specification of a hypergraph defined by (Gallo et al., 1993) will be redefined for the access control with $\mathcal{HGAC}$.

A hypergraph as defined by (Gallo et al., 1993) is a graph $G_{hyp} = (V, E)$ with the set of nodes $V = \{v_1, v_2, ..., v_n\}$ and the set of hyperedges $E = \{E_1, E_2, ..., E_m\}$. A directed hyperedge $E_i = (S, Z)$ consists of arbitrary non-empty sets of *start nodes* $S \subseteq V$ and *target nodes* $Z \subseteq V$.

$\mathcal{HGAC}$ redefines the introduced hypergraph definition by relations on relations (cf. section 3.2). It uses language expressions to declare subjects (cf. section 4).

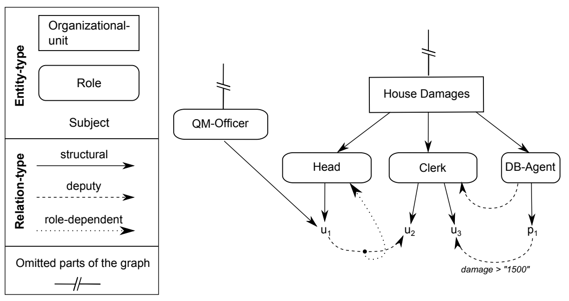


Figure 3: Excerpt of an Organizational Model of an Insurance Company.

3.1 Organizational Model

Structural Relations

Figure 3 shows an example model of an insurance company[7]. The model consists of the department *House Damages*, the subjects u_1, u_2, u_3 (human subjects) and p_1 (automatic subjects) with their functional units *Head*, *Clerk* and *DB-Agent*. The subject u_1 is also working as *QM-Officer* a position within another department.

Organizational Relations

Beside the structural relations, the company's model contains further relations – organizational relations[8] (e.g. deputy, supervisor and reporting relations) that interconnect entities. In this example, the *Head* u_1 has a deputy u_2 if u_1 acts as *Head* of the department *House Damages*[9]. If u_1 acts as *QM-Officer*, then u_2 is not a possible deputy.

In case the *DB-Agent* p_1 is unavailable, a constrained deputy relation to u_3 is evaluated at

[4]The context in which a subject acts (e.g. "purchase" in a workflow).

[5]Access rights are assigned using attributes of e.g. a subject (like "Hiring Year" > 2).

[6]Access rights defined by parameters passed from an application system. If for example the "damage" in an insurance case amounts to 200000, only subjects fulfilling this are responsible.

[7]The metamodel is described in (Lawall et al., 2014b).

[8]In (Lawall et al., 2012), (Lawall et al., 2013a) and (Lawall et al., 2013b), the complete metamodel and formal language including, i.a. constraints, are specified.

[9]It is possible to restrict any and all organizational relations to be role-dependent.

first. If the parameter-based[10] predicate (`damage > "1500"`) of the relation is fulfilled, it transfers all access rights of p_1 to u_3. In this scenario, the subject u_3 is a *Clerk* responsible for expensive insurance cases. If u_3 and p_1 are simultaneously unavailable, the general deputy relation between *DB-Agent* and *Clerk* is evaluated. The result set consists of subject u_2 which is then the deputy. u_3 is not an element of this set because of unavailability. This algorithm represents a prioritization mechanism.

Another variant to restrict the validity of relations are context-sensitive constraints (not included in the example). A constraint assigned to a deputy relation of a subject is fulfilled if the context is identical to the context provided by an application system.

3.2 Formal Specification

The start node is substituted in the formal specification as $\alpha := \textit{Permission}$.

The access control hypergraph $G_{Perm} = (V, E)$ is defined as:

- The set of nodes $V = \{\alpha, o_1, ..., o_l\}$ with $o_g \in O \wedge g = 1, ..., l$
  - The node α is the reference node of all access right definitions.
  - O the set of objects which are assigned to rights.
- The set of edges $E = \{r_j^h \mid r_j^h \in \xi_j \wedge h = 1, ..., d\}$ consists of ternary relations r_j^h (j: relation-types for rights, d: number of relations per relation-type)
  - $\mathcal{R}$: set of relation-types of rights
  - $\xi_j \in \mathcal{R}$: set of relations of a specific relation-type of a right
  - $\forall r_j^h (r_j^h \in \xi_j) \colon r_j^h \subseteq (\{\alpha\} \cup \xi_j) \times (O \cup \xi_j) \times \mathcal{L}$:[11]
    * The element $\{\alpha\} \cup \xi_j$ of the tuple declares the start of an edge in G_{Perm}. $\alpha \in V$ is the first start node of every relation of a relation-type j of an hyperedge. Afterwards, arbitrary relations $r_j^b \in \xi_j$ with $b = 1, ..., |\xi_j| - 1$ can be the start of an edge of the same relation-type j.
    * $O \cup \xi_j$ defines the end of an edge in the graph. This can be a concrete object $o \in (O \subset V)$ or a set of relations of a relation-type ξ_j.
    * A language expression $\mathcal{L}$ is a valid element of language $L(G)$[12]. The empty word ε is included in the language $L(G)$ as well.
  - $f_{\mathcal{L}} \colon r_j^h \to \mathcal{L}$ extracts the language expression $\mathcal{L}$ assigned to the hyper-relation r_j^h.
- The subjects $\mathcal{S}$ result from evaluating the language expressions $\mathcal{L}$ on the model.
  - $\mathcal{L} \Rightarrow^* \mathcal{S}_{part} \subseteq \mathcal{S}$ defines the result set $\mathcal{S}_{part}$ of the language expression $\mathcal{L}$. The symbol $\Rightarrow^*$ indicates the resulting subjects of the language expression $\mathcal{L}$ by traversing the organizational model. The traversal algorithms are formalized in (Lawall et al., 2014a) and (Lawall et al., 2014c).
  - $\mathcal{S}_{path} = \bigcup_{p=1}^{e} \mathcal{S}_{part}$ with e equals the path length, defines the set of all subjects assigned to access right j on the path $P_j^{o \in O}$. This path starts in α and ends in $o \in O$. All language expressions $\mathcal{L}$ of the relations r_j^h on $P_j^{o \in O}$ are concatenated with `OR`[13]. The resulting expression is evaluated on the model to get the subjects $\mathcal{S}_{path}$.

The set of *all* subjects assigned to an access right j for an object $o \in O$ can be evaluated differently. It is possible to compare all paths of a right j related to the object o.

Starting the traversal in object o is more efficient because an unnecessary evaluation of paths containing $O \setminus o$ is excluded. The direction of the concatenation of the language expressions is in “reverse” order. The reverse and forward concatenation of language expressions results in identical subjects.

3.3 Definition of Access Rights

An example (cf. fig. 4) configuration of the hypergraph $G_{Perm} = (V, E)$ is:

- $O = \{f1, f2, f3, p1, p2, p3\}$: the set of objects containing files and processes
- $V = \{\alpha, f1, f2, f3, p1, p2, p3\}$
- $E = \{r_{read}^1, r_{write}^1, r_{write}^2, r_{write}^3, r_{write}^4, r_{write}^5, r_{exec}^1, r_{exec}^2, r_{exec}^3, r_{exec}^4, r_{exec}^5\}$
  - $\mathcal{R} = \{\mathcal{READ}, \mathcal{WRITE}, \mathcal{EXECUTE}\}$
  - $\xi_{read} = \{r_{read}^1\}$
  - $\xi_{write} = \{r_{write}^1, r_{write}^2, r_{write}^3, r_{write}^4, r_{write}^5\}$
  - $\xi_{exec} = \{r_{exec}^1, r_{exec}^2, r_{exec}^3, r_{exec}^4, r_{exec}^5\}$

[10]If a value is handed from an application system to the organizational model via the formal language, the expression is called parameter-based.

[11]The relation r_j^1 includes **always** the “Permission” (α) node.

[12]The syntax and semantic of the language is defined in (Lawall et al., 2013a).

[13]The empty word ε is excluded from this concatenation.

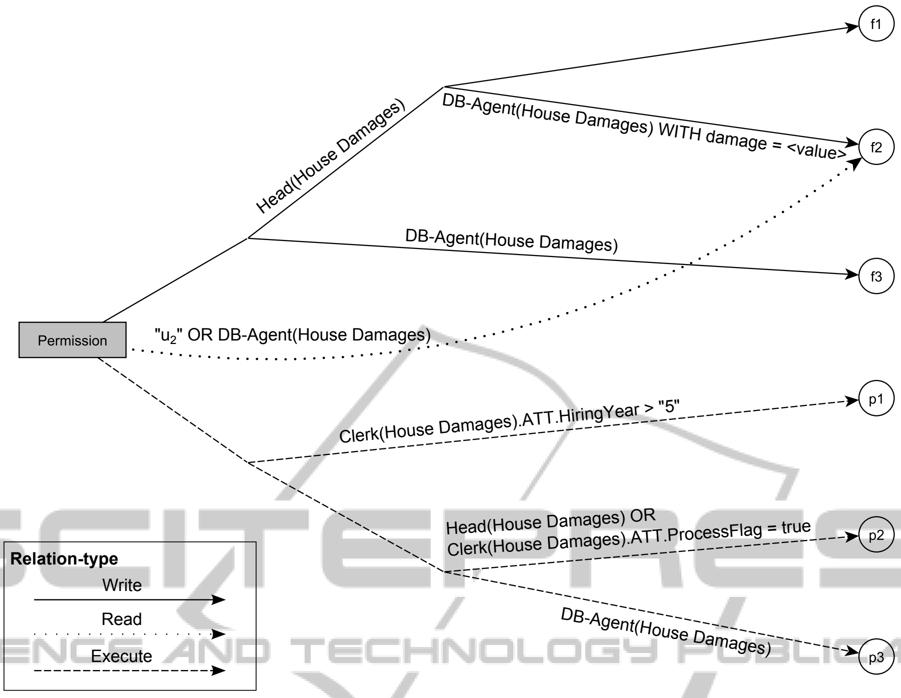

f1
f2
f3
p1
p2
p3
Permission
Head(House Damages)
DB-Agent(House Damages) WITH damage = <value>
DB-Agent(House Damages)
"u2" OR DB-Agent(House Damages)
Clerk(House Damages).ATT.HiringYear > "5"
Head(House Damages) OR
Clerk(House Damages).ATT.ProcessFlag = true
DB-Agent(House Damages)
Relation-type
Write
Read
Execute

Figure 4: Access Rights defined by using a Hypergraph and Formal Expressions.

- $r^1_{read} = (\{\alpha\}, \{f2\},$ "u2" OR DB-Agent (House Damages))
- $r^1_{write} = (\{\alpha\}, \{r^2_{write}, r^3_{write}\}, \varepsilon)$
- $r^2_{write} = (\{r^1_{write}\}, \{f3\},$ DB-Agent(House Damages))
- $r^3_{write} = (\{r^1_{write}\}, \{r^4_{write}, r^5_{write}\},$Head(House Damages))
- $r^4_{write} = (\{r^3_{write}\}, \{f2\},$ DB-Agent(House Damages) WITH damage = "2000")
- $r^5_{write} = (\{r^3_{write}\}, \{f1\}, \varepsilon)$
- $r^1_{exec} = (\{\alpha\}, \{r^2_{exec}, r^3_{exec}\}, \varepsilon)$
- $r^2_{exec} = (\{r^1_{exec}\}, \{p1\},$ Clerk(House Damages).ATT.HiringYear > "5")
- $r^3_{exec} = (\{r^1_{exec}\}, \{r^4_{exec}, r^5_{exec}\}, \varepsilon)$
- $r^4_{exec} = (\{r^3_{exec}\}, \{p2\},$Head(House Damages) OR Clerk(House Damages).ATT.Processflag = "true")
- $r^5_{exec} = (\{r^3_{exec}\}, \{p3\},$DB-Agent(House Damages))

• The set of subjects $\mathcal{S} = \{u_1, u_2, u_3, p_1\}$ (human and automatic)[14] (cf. fig. 3):
  - $f_{\mathcal{L}}(r^1_{read}) =$ "u2" OR DB-Agent(House Damages) $\Rightarrow^* \mathcal{S}_{part} = \{u_2, p_1\}$
  - $f_{\mathcal{L}}(r^2_{write}) = f_{\mathcal{L}}(r^5_{exec}) =$ DB-Agent(House Damages) $\Rightarrow^* \mathcal{S}_{part} = \{p_1\}$
  - $f_{\mathcal{L}}(r^3_{write}) =$ Head(House Damages) $\Rightarrow^* \mathcal{S}_{part} = \{u_1\}$
  - $f_{\mathcal{L}}(r^4_{write}) =$ DB-Agent(House Damages) WITH damage = "2000" $\Rightarrow^* \mathcal{S}_{part} = \{p_1\}$
  - $f_{\mathcal{L}}(r^2_{exec}) =$ Clerk(House Damages).ATT.HiringYear > "5" $\Rightarrow^* \mathcal{S}_{part} = \{u_2\}$[15]
  - $f_{\mathcal{L}}(r^4_{exec}) =$ Head(House Damages) OR Clerk(House Damages).ATT.Processflag = "true" $\Rightarrow^* \mathcal{S}_{part} = \{u_1, u_3\}$[16]
  - $f_{\mathcal{L}}(r^1_{write}) = f_{\mathcal{L}}(r^5_{write}) = f_{\mathcal{L}}(r^1_{exec}) = f_{\mathcal{L}}(r^3_{exec}) = \varepsilon \Rightarrow^* \mathcal{S}_{part} = \emptyset$

• The set of all subjects assigned to access rights on an object $o \in O$:
  - Access right $\mathcal{WRITE}$ to an object $f1$ is assigned to subject u_1:
    * Path $P^{f1}_{\mathcal{WRITE}} = (r^5_{write}, r^3_{write}, r^1_{write})$
    * $\mathcal{S}_{path} = \emptyset \cup \{u_1\} \cup \emptyset = \{u_1\}$
  - Access right $\mathcal{WRITE}$ to an object $f2$ is assigned to subjects u_1, p_1 and access right

[14]The following results can be different in cases of absence of subjects depending on deputy relations.

[15]u_3 is not included in the resulting set of subjects because the attribute-based predicate is not fulfilled.

[16]u_3 is included in the resulting set of subjects because the value of his attribute fulfills the attribute-based predicate.

$\mathcal{READ}$ to u_2, p_1:

$\mathcal{WRITE}$:

- Path $P^{f2}_{\mathcal{WRITE}} = \left(r^4_{write}, r^3_{write}, r^1_{write}\right)$
- $S_{path} = \{p_1\} \cup \{u_1\} \cup \emptyset = \{u_1, p_1\}$

$\mathcal{READ}$:

- Path $P^{f2}_{\mathcal{READ}} = \left(r^1_{read}\right)$
- $S_{path} = \{u_2, p_1\}$

– Access right $\mathcal{WRITE}$ to an object $f3$ is assigned to subject p_1:

- Path $P^{f3}_{\mathcal{WRITE}} = \left(r^2_{write}, r^1_{write}\right)$
- $S_{path} = \{p_1\} \cup \emptyset = \{p_1\}$

– Access right $\mathcal{EXECUTE}$ to an object $p1$ is assigned to subject u_2:

- Path $P^{p1}_{\mathcal{EXECUTE}} = \left(r^2_{exec}, r^1_{exec}\right)$
- $S_{path} = \{u_2\} \cup \emptyset = \{u_2\}$

– Access right $\mathcal{EXECUTE}$ to an object $p2$ is assigned to subjects u_1, u_3:

- Path $P^{p2}_{\mathcal{EXECUTE}} = \left(r^4_{exec}, r^3_{exec}, r^1_{exec}\right)$
- $S_{path} = \{u_1, u_3\} \cup \emptyset \cup \emptyset = \{u_1, u_3\}$

– Access right $\mathcal{EXECUTE}$ to an object $p3$ is assigned to subject p_1:

- Path $P^{p3}_{\mathcal{EXECUTE}} = \left(r^5_{exec}, r^3_{exec}, r^1_{exec}\right)$
- $S_{path} = \{p_1\} \cup \emptyset \cup \emptyset = \{p_1\}$

The definition of a specific access right using (hyper-) relations and language expressions in conjunction with the organizational model simplifies the maintenance if the company's organization changes. This is established by the logically centralized organizational model and the language expressions that are stored in the hypergraph.

The maintenance effort for applying company policies decreases with $\mathcal{HGAC}$. In the example, the path $P^{f2}_{\mathcal{WRITE}} = \left(r^4_{write}, r^3_{write}, r^1_{write}\right)$ defines the access right $\mathcal{WRITE}$ assigned to object $f2$. The subjects resulting from the language expression `DB-Agent(House Damages) WITH damage = "2000"` assigned to the relation r^4_{write} is exclusively valid for object $f2$.
The relation r^3_{write} with the language expression `Head(House Damages)` is valid for the objects $f1$ and $f2$. The expression does not have to be stored twice. Thus, $\mathcal{HGAC}$ provides a mechanism to avoid redundancies.

Access rights for new objects in application systems can be integrated at any point of the access right hypergraph. The user can reuse previously defined relations and assigned expressions. The user connects the new relation to the hypergraph. If needed, he assigns a new language expression to get the appropriate set of subjects that have access to the new object.

For example, a user adds an object $f4$ (e.g. financial report for the department House Damages) to an application system. The policy for this object implies that the Head and Clerks of the department House Damages can access it, but not the DB-Agent. Therefore, the user instantiates a relation of the type $\mathcal{WRITE}$ and assigns the expression `Clerk(House Damages)`. This new relation $r^6_{write} = (\{r^3_{write}\}, \{f4\},$ `Clerk(House Damages)`$)$ originates in r^3_{write}[17] and ends in object $f4$. Thus, the policy is defined by reusing existing parts. This avoids redundancies.

For clarification, all redundant expressions (e.g. `DB-Agent (House Damages)`) shown in figure 4 are "redundantly" written to simplify reading. Such recurring expressions can be stored in macros, so that changes in the macro affect all (hyper-) relations that store this macro. Macros avoid redundancies in the definition of access rights respectively policies with $\mathcal{HGAC}$.

4 DECLARATIVE LANGUAGE

The domain-specific language $L(G)$ is defined by the context-free grammar $G = (N, \Sigma, P, S_G)$ with the set of non-terminals N, the alphabet Σ with $N \cap \Sigma = \emptyset$, the production rules P and the start symbol S_G with $S_G \in N$ (cf. (Fowler, 2010)). Each production rule has the format $l \to r$ with $l \in N$ and $r \in (N \cup \Sigma)^*$.

4.1 Syntax

Non-terminals that expand only to one specific sequence of terminals (keywords) are represented as e.g. '`NOT`', '`WITH`'.

The grammar G_1 for defining *queries* is a tuple of:

- The set of non-terminals $N_1 = \{start, query, actor, funits, funit, oudef, ounits, ounit, relationTokens, withParams, contextDefinition, attConstraints, kcv, parameter, kvp, id, string\}$
- The alphabet of terminals $\Sigma_1 = \{$'`a`','`b`',...,'`z`','`A`','`B`',...,'`Z`','`ä`','`ü`','`ö`','`Ä`','`Ü`','`Ö`', '`0`','`1`',...,'`9`','`_`','`-`','`(`','`)`','`,`',' ','`.`','`*`','`=`','$<$','$>$'$\}$[18]
- The set of production rules P_1[19]

[17] r^3_{write} includes `Head(House Damages)`.

[18] The terminals derived from the non-terminal *string* are also included.

[19] Meaning of meta-symbols: ? means 0 or 1 and * means 0 to ∞ occurrences.

$start \rightarrow query \mid query\ logic\ query \mid \varepsilon$
$query \rightarrow actor \mid actor\ \text{‘}\texttt{AS}\text{’}\ funits$
$query \rightarrow query\ \text{‘}\texttt{NOT}\text{’}\ query$
$query \rightarrow query\ \text{‘}\texttt{FALLBACKTO}\text{’}\ query$
$query \rightarrow query\ \text{‘}\texttt{WITH}\text{’}\ withParams$
$query \rightarrow funits\ \text{‘}\texttt{(}\text{’}oudef\text{‘}\texttt{)}\text{’}$
$query \rightarrow relationTokens\ \text{‘}\texttt{(}\text{’}query\text{‘}\texttt{)}\text{’}$
$query \rightarrow \text{‘}\texttt{(}\text{’}query\ logic\ query\text{‘}\texttt{)}\text{’}$
$query \rightarrow \text{‘}\texttt{(}\text{’}query\text{‘}\texttt{).}\text{’}attConstraints$
$actor \rightarrow \text{‘}\texttt{*}\text{’} \mid id \mid string$
$funits \rightarrow funit \mid \text{‘}\texttt{(}\text{’}funit\ logic\ funit\text{‘}\texttt{)}\text{’}$
$funit \rightarrow \text{‘}\texttt{*}\text{’} \mid id \mid string$
$oudef \rightarrow ounit \mid ounits\ logic\ ounits$
$ounits \rightarrow ounit \mid \text{‘}\texttt{(}\text{’}ounits\ logic\ ounits\text{‘}\texttt{)}\text{’}$
$ounit \rightarrow \text{‘}\texttt{*}\text{’} \mid id \mid string \mid ounit\ \text{‘}\texttt{SUBS}\text{’}$
$relationTokens \rightarrow (\text{‘}\texttt{ALL}\text{’} \mid \text{‘}\texttt{ANY}\text{’})?\ id\ (\text{‘}\texttt{OF}\text{’} \mid \text{‘}\texttt{TO}\text{’})$
$withParams \rightarrow contextDefinition \mid parameter \mid withParams\ \text{‘,’}\ withParams$
$contextDefinition \rightarrow \text{‘}\texttt{CONTEXT=}\text{’}\ context\ (\text{‘,’}\ context)*$
$attConstraints \rightarrow \text{‘}\texttt{ATT.}\text{’}\ kcv$
$kcv \rightarrow id\ comp\ string \mid \text{‘}\texttt{(}\text{’}kcv\ logic\ kcv\text{‘}\texttt{)}\text{’}$
$parameter \rightarrow kvp\ (\text{‘,’}\ kvp)*$
$kvp \rightarrow id\ \text{‘=’}\ string$
$logic \rightarrow \text{‘}\texttt{AND}\text{’} \mid \text{‘}\texttt{OR}\text{’}$
$comp \rightarrow (\text{‘}\texttt{=}\text{’} \mid \text{‘}\texttt{<=}\text{’} \mid \text{‘}\texttt{>=}\text{’} \mid \text{‘}\texttt{<}\text{’} \mid \text{‘}\texttt{>}\text{’} \mid \text{‘}\texttt{!=}\text{’})$
$id \rightarrow ([\text{‘}\texttt{a}\text{’-‘}\texttt{z}\text{’,‘}\texttt{A}\text{’-‘}\texttt{Z}\text{’}] \mid \text{‘}\texttt{\_}\text{’} \mid \text{‘}\texttt{Ä}\text{’} \mid \text{‘}\texttt{ä}\text{’} \mid \text{‘}\texttt{Ü}\text{’} \mid \text{‘}\texttt{ü}\text{’} \mid \text{‘}\texttt{Ö}\text{’} \mid \text{‘}\texttt{ö}\text{’})\ ([\text{‘}\texttt{a}\text{’-‘}\texttt{z}\text{’,‘}\texttt{A}\text{’-‘}\texttt{Z}\text{’}] \mid \text{‘}\texttt{Ä}\text{’} \mid \text{‘}\texttt{ä}\text{’} \mid \text{‘}\texttt{Ü}\text{’} \mid \text{‘}\texttt{ü}\text{’} \mid \text{‘}\texttt{Ö}\text{’} \mid \text{‘}\texttt{ö}\text{’} \mid [0-9] \mid \text{‘}\texttt{\_}\text{’} \mid \text{‘}\texttt{-}\text{’})^*$
$string \rightarrow \text{‘}\texttt{"}\text{’}\ id\ \text{‘}\texttt{"}\text{’}$

- The set of start symbols $S_{G_1} = \{start\}$

The grammar G_2 for defining *predicates on relations* is a tuple of:

- The set of non-terminals $N_2 = \{internal, relPred, parameteratt, context, parameter, attribute, kcv, logic, id\}$
- The alphabet of terminals $\Sigma_2 = \Sigma_1$
- The set of production rules P_2[20]

$internal \rightarrow relPred \mid relPred\ logic\ relPred \mid \varepsilon$
$relPred \rightarrow context \mid parameteratt \mid \text{‘}\texttt{(}\text{’}relPred\text{‘}\texttt{)}\text{’} \mid \text{‘}\texttt{(}\text{’}relPred\ logic\ relPred\text{‘}\texttt{)}\text{’}$
$parameteratt \rightarrow parameter \mid attribute \mid \text{‘}\texttt{(}\text{’}parameteratt\ logic\ parameteratt\text{‘}\texttt{)}\text{’}$
$context \rightarrow id \mid \text{‘}\texttt{(}\text{’}context\ logic\ context\text{‘}\texttt{)}\text{’}$
$parameter \rightarrow kcv \mid \text{‘}\texttt{(}\text{’}parameter\ logic\ parameter\text{‘}\texttt{)}\text{’}$
$attribute \rightarrow \text{‘}\texttt{ATT.}\text{’}\ kcv \mid \text{‘}\texttt{(}\text{’}attribute\ logic\ attribute\text{‘}\texttt{)}\text{’}$

- The set of start symbols $S_{G_2} = \{internal\}$

The grammar G is the result of the union of grammars G_1 and G_2[21]. This equals $G = \{N_1 \cup N_2, \Sigma_1, P_1 \cup P_2 \cup \{s_G \rightarrow start \mid internal\}, \{s_G\}\}$ A language expression $\mathcal{L}$ is syntactically correct if $\mathcal{L}$ is derivable starting from the set of start symbols of the language: $L(G) = \left\{\mathcal{L} \in \Sigma^* \mid S_G \leadsto^*_G \mathcal{L}\right\}$. The bottom-up approach for the syntactical evaluation is also possible.

4.2 Semantics

Language expressions are formulated *within* or *outside* of the organizational model. *Within* means that the expression is inside of an organizational model and represents a *predicate* assigned to a relation. The grammar for the definition of predicates is G_2.

The syntax for *queries* is defined by the grammar G_1. They are the *outside* perspective. Application systems pass language expressions to the organizational model (via organizational server, cf. section 6, fig. 6). The expression is then evaluated on the organizational model which yields subjects.

The semantics of the domain-specific language $L(G)$ is described informally for brevity. The semantics of *queries* that are formulated to get the appropriate set of subjects is the following[22] – $L(G_1)$:

- **Based on Structural Relations** are queries describing a concrete subject, e.g. “`u1`”, subjects having a specific role in a specific organizational-unit e.g. `Clerk(House Damages)`, or subjects having a specific role in any organizational-unit e.g. `Clerk(*)`.
- **Separation of Duty** is expressed by the `NOT`-clause mostly used in workflow management systems, e.g. `SUPERVISOR OF(<initiator>) NOT <initiator>`, to prevent the initiator of a process from approving his own request for purchase.
- **Prioritization** defines primary candidates and an alternative to fall back to. If the set of primary candidates is empty, the second query is evaluated, e.g. `Clerk(House Damages) FALLBACKTO Head(House Damages)`.

[20]Production rules for non-terminals *kcv*, *logic* and *id* correspond to those in P_1.

[21]Union of context-free grammars according to (Hoffmann, 2011).

[22]Composite access rights can be defined by the concatenation of queries with logical `AND` and `OR`.

- **Parameter Passing** is done using `WITH`. Application systems pass parameters to the organizational model, e.g. `DB-Agent(House Damages) WITH damage = "1000"`.
- **Acting Role** describes the role in which a subject acts. This is used to decide if role-dependent relations are valid, e.g. the deputy relation between $u1$ and $u2$. This deputyship is only valid if $u1$ acts as $Head$ (cf. section 3.1). There are different possibilities to formulate such queries. The implicit enumeration of roles e.g. `Head(House Damages)` specifies the role *Head*. The explicit variant to pass a role to the organizational model when directly naming a subject is possible with e.g. `u1 AS Head`.
- **Subjects restricted by Attributes** are subjects that fulfill the attribute constraints. The expression `Clerk(House Damages).ATT.HiringYear > "2"` describes the subjects that have been on the job for more than two years.
- **Based on organizational Relations** that are deputyship, supervision and reporting dependencies. This can also be used for audits. If, for example, all possible deputies have to be listed, the expression `ANY DEPUTY OF(p1)` can be used. The possible subjects are $u2$ and $u3$, stemming from the predicated and not predicated deputy relations from $p1$ and *DB-Agent*. `ANY` ignores predicates (e.g. `damage > "1500"` or role-dependent predicates). `ALL` indicates that the relation is to be followed transitively.

The semantic of *predicates* on relations that are used to declare organizational circumstances are[23] – $L(G_2)$:

- **General Validity** means that if no predicate is assigned to a relation (ε) it is always valid, e.g. the deputy relation between the roles *DB-Agent* and *Clerk*.
- **Based on the Current Context the Subject is supposed to Act in**, specific relations can be valid or invalid. This has consequences for organizational regulations, e.g. $u2$ can only be $u1$'s deputy, if $u1$ acts as *Head*, not if he acts as *QM-Officer*. Another context can be a "purchase". A predicate is e.g. `purchase.damage > "1500"`. The deputyship between $p1$ and $u3$ is dependent on the context "purchase".
- **Based on Attributes of the Subjects**, subjects can be filtered from the result set. Relations on the path do not change their validity by this mechanism.
- **Based on Parameters** from application systems, the relations may be valid or invalid. This directly influences the traversal of relations within the organizational model.

[23]Predicates can be based on a combination of context, attributes and parameters.

Discussion

The novel approach $\mathcal{HGAC}$ reduces the maintenance effort to zero. If subjects join, move or leave the company, no changes have to be done concerning access right assignment. The access rights are immediately and automatically consistent.

On the one hand, this is enabled by the organizational (meta-) model of the company's circumstances, cf. section 3.1, and on the other hand by the declarative language, cf. chapter 4. The policies are formulated in language expressions that describe the requirements for access.

The expressions declare queries for policies / access rights in application systems. They are based on organizational structures (entities and relations) and consider properties of subjects (attributes).

Additionally, the language is used to define predicates. They are used for policies that are formulated on relations in the organizational model. The conjunction of queries and predicates is a powerful tool for defining policies.

Characteristic technical roles, as needed in RBAC, to define the needed policies / access rights are obsolete. Policies are described by language expressions and structured using hyperedges in $\mathcal{HGAC}$. Thus, changes – property changes (e.g. name, hiring year, salary, etc.) and relation changes (e.g. join, move, leave, new supervisor or deputy relation, etc.) – concerning subjects do not affect the access right and policy definitions.

If, for example, a subject $s \in \mathcal{S}$ joins $j(s)$, moves $m(s)$ or leaves $l(s)$ the company, the effort[24] in $\mathcal{HGAC}$ maintaining access rights in the application systems is:

$$\mathcal{HGAC}_{j(s),m(s),l(s)} = 0 \qquad (4)$$

The only maintenance effort is in the organizational model, cf. section 5.

Structuring access rights is limited in ACM and RBAC approaches. The access rights can be structured hierarchically, e.g. for directory rights and contained directories and files. This is an object-centered view. The objects are generally structured. $\mathcal{HGAC}$ favors an operator-centered view. The operators (i.a.

[24]The effort is identical for relation changes, e.g. adding a supervisor relation in the organizational model.

Table 1: Access Rights on Object “Project X” by Role.

Rights $\mathcal{R}$	**Role**	**Subjects in Role**
f,m,e,l,r,w	$Role_A$	16
f,m,e,l,r,w	$Role_B$	13
f,m,e,l,r,w	$Role_C$	42
f,m,e,l,r,w	$Role_D$	4
e,l,r	$Role_E$	13

Table 2: Access Rights on Object “International” by Role.

Rights $\mathcal{R}$	**Role**	**Subjects in Role**
f,m,e,l,r,w	$Role_A$	16
e,l,r	$Role_F$	193
e,l,r	$Role_G$	150
f,m,e,l,r,w	$Role_H$	64
e,l,r	$Role_I$	3983
e,l,r	$Role_J$	172
e,l,r	$Role_K$	305
e,l,r	$Role_L$	178
e,l,r	$Role_M$	65
m,e,l,r,w	$AD-HOC$	5

read, write) are structured and the subjects are declared by the language expressions. This makes it easy to find all subjects that have a specific access right to a specific object.

5 CASE STUDY

In order to validate and compare the approach, a case study was conducted. Two objects with access rights assigned using RBAC were evaluated. These objects are directories on a file server:

- “Project X” (cf. table 1) is an arbitrarily selected directory of a research project.
- “International” (cf. table 2) is a directory containing resources for academic international affairs.

Tables 1 and 2 show the permissions that are assigned to different roles for the objects. They also list the number of subjects that are assigned to the individual roles. The set $\mathcal{R} = \{full\ access, modify, execute, list, read, write\}$ defines the access rights that can be assigned. For brevity, they are denoted in the tables as $\{f,m,e,l,r,w\}$ correspondingly.

As can be seen from the amount of subjects, an ACM approach would not be practical with almost 4000 subjects, even for such a small number of objects[25].

A detailed look at the subject assignments revealed a number of inconsistencies:

- Subjects occur multiple times in different roles.
- Subjects are assigned to roles that have more rights than the subject should have. A student assistant had the role of a researcher.
- The ad-hoc role $AD-HOC$ is a technical role specific to the object “International”. It is not used anywhere else and contains a reference to a subject that does no longer exist in the directory server.
- There exist a number of pseudo-subjects, such as test accounts that allow system administrators to impersonate members of specific roles.
- Technical roles (subjects with the same rights) and organizational roles (subjects with the same job position) are mixed arbitrarily.

This list of discrepancies illustrates how error-prone the maintenance of join, move and leave operations is in RBAC. For each operation, all affected role assignments have to be maintained. In $\mathcal{HGAC}$, these operations have to be performed *once* in the organizational model.

Figure 5 shows the hypergraph in $\mathcal{HGAC}$ that is equivalent to the representation in RBAC. For clarity, the different sets of rights $\{e,l,r\}$, $\{f,m,e,l,r,w\}$ and $\{m,e,l,r,w\}$ are represented as one relation-type each in the depiction. The actual hypergraph contains a relation-type per access right.

The key of the representation of an RBAC model in $\mathcal{HGAC}$ is the formulation of the roles as language expressions:

- $Role_A$ represents the IT-administrators of the University: `Admin(University)`
- $Role_B$, $Role_C$ and $Role_D$ are technical roles for members of the Research Department: `Member(Research Department)`
- $Role_E$ are employees of the IT-Infrastructure department: `Member(IT-Infrastructure)`
- $Role_F$ and $Role_G$ encompass lecturers of the university: `Lecturer(*)`.
- $Role_H$ are employees of the Datacenter department: `Member(Datacenter)`.
- $Role_I$ $Role_J$ and $Role_K$ represent different types of students, e.g. external and internal students. As they appear together, they can be represented as `Student(*)`.
- $Role_L$ represents all administrative employees of the university, `*(Administration)`.
- $Role_M$ are research assistants of the University, as can be described as `Research Assistant(University)`.

[25]subjects * objects * access rights $\approx 4000 \cdot 2 \cdot 6$

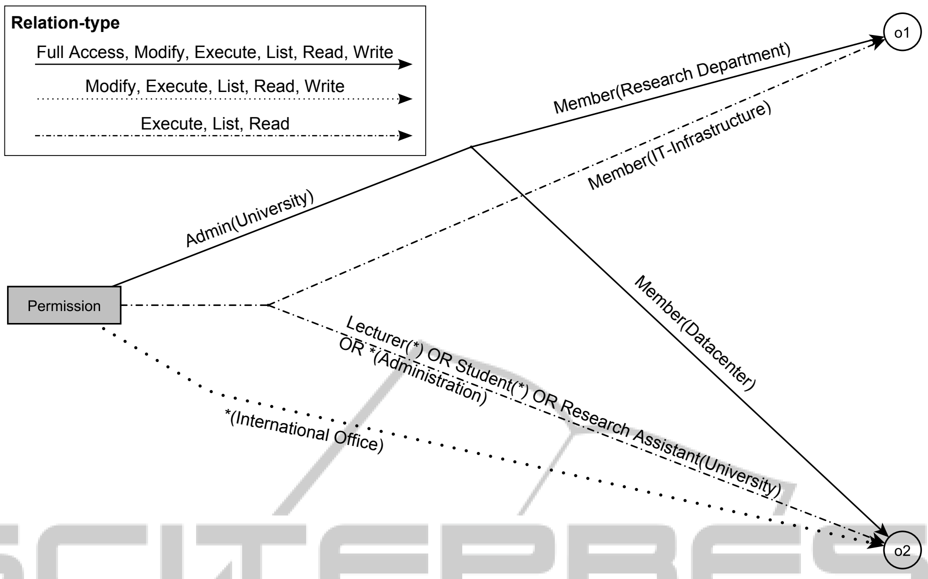

Relation-type
Full Access, Modify, Execute, List, Read, Write
Modify, Execute, List, Read, Write
Execute, List, Read
Permission
Admin(University)
Member(Research Department)
Member(IT-Infrastructure)
Member(Datacenter)
Lecturer(*) OR Student(*) OR Research Assistant(University)
OR *(Administration)
*(International Office)
o1
o2

Figure 5: Access Rights of the Case Study in $\mathcal{HGAC}$.

- The special role $AD-HOC$ contains all employees of the International Office of the university. The expression `*(International Office)` describes them.

6 CONCLUSION

The approaches ACM and RBAC require extensive *maintenance effort* (cf. equations (1), (2) and (3)) to accommodate fluctuating subjects in the company. This effort is erased in $\mathcal{HGAC}$ (cf. equation (4)). The language expressions declaring policies / access rights are more stable over time than total enumeration used by ACM and RBAC. Access rights are defined in $\mathcal{HGAC}$ in a descriptive manner. The description can be based on organizational structures (i.a. departments, job functions, supervisor relations) and properties (i.a. name, hiring year, salary). In addition, parameters from application systems or different contexts can be decisive. The contexts are defined by the nature of necessary tasks or of common resources. The organizational model includes the definition of responsible subjects for the administration of the organizational model itself. This decreases the workload of administrators and distributes the work to the subjects that maintain the organizational model.

The subject that joins, moves or leaves the company causes maintenance effort in all application systems respectively applications which is prone to error. As a consequence, the definition of *consistent access rights* that conform to reality is facilitated.

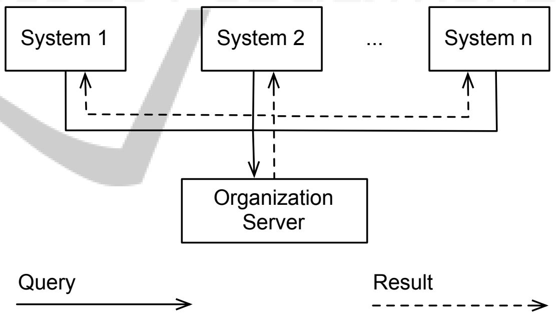


Figure 6: Organizational Server connected with Application Systems adapted from (Lawall et al., 2014b).

The concept of $\mathcal{HGAC}$ in conjunction with an organizational server solves the afore-mentioned problems (cf. figure 6). The systems hand the language expressions (query) to the organizational server holding the organizational model of the company. The expressions are evaluated on the model and the set of authorized subjects are handed back to the system (result). All connected systems are immediately in synchronization with organizational facts if the new organizational conditions are modeled in the organizational server. This makes access rights consistent over various application systems.

The focus for future research is the extension of the language to overcome problems resulting from renaming organizational entities. A macro-like mechanism will be examined to have a single point of maintenance for expressions. Macros remedy the redundant storage of identical expressions. They are references to expressions. Only expressions referenced by macros have to be changed.

REFERENCES

Benantar, M. (2006). *Access Control Systems: Security, Identity Management and Trust Models*. Access Control Systems: Security, Identity Management and Trust Models. Springer.

Chen, L. (2011). *Analyzing and Developing Role-Based Access Control Models*. PhD thesis, University of London.

Chen, Y. and Zhang, L. (2011). Research on role-based dynamic access control. In *Proceedings of the 2011 iConference*, iConference '11, pages 657–660, New York, NY, USA. ACM.

Ferraiolo, D., Kuhn, D., and Chandramouli, R. (2003). *Role-based Access Control*. Artech House computer security series. Artech House.

Ferraiolo, D. F., Barkley, J. F., and Kuhn, D. R. (1999). A role-based access control model and reference implementation within a corporate intranet. *ACM Trans. Inf. Syst. Secur.*, 2:34–64.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4:224–274.

Ferrari, E. (2010). *Access Control in Data Management Systems*. Synthesis lectures on data management. Morgan & Claypool.

Fowler, M. (2010). *Domain-Specific Languages*. Addison-Wesley Professional.

Gallo, G., Longo, G., Pallottino, S., and Nguyen, S. (1993). Directed Hypergraphs and Applications. *Discrete Appl. Math.*, 42(2-3):177–201.

Graham, G. S. and Denning, P. J. (1972). Protection: Principles and Practice. In *Proceedings of the May 16-18, 1972, Spring Joint Computer Conference*, AFIPS '72 (Spring), pages 417–429, New York, NY, USA. ACM.

Hoffmann, D. W. (2011). *Theoretische Informatik*. München: Carl Hanser, München, 2. edition.

Knorr, K. (2000). Dynamic access control through Petri net workflows. In *Computer Security Applications, 2000. ACSAC '00. 16th Annual Conference*, pages 159–167.

Krcmar, H. (2010). *Informationsmanagement*. Springer, Berlin; Heidelberg.

Lawall, A., Schaller, T., and Reichelt, D. (2012). An Approach towards Subject-Oriented Access Control. In *S-BPM ONE 2012*, pages 33–42, Heidelberg. Springer-Verlag.

Lawall, A., Schaller, T., and Reichelt, D. (2013a). Integration of Dynamic Role Resolution within the S-BPM Approach. In *S-BPM ONE 2013*, pages 21–33, Heidelberg. Springer.

Lawall, A., Schaller, T., and Reichelt, D. (2013b). Who Does What – Comparison of Approaches for the Definition of Agents in Workflows. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 74–77.

Lawall, A., Schaller, T., and Reichelt, D. (2014a). Cross-Organizational and Context-Sensitive Modeling of Organizational Dependencies in C-ORG. In *S-BPM ONE (Scientific Research)*, pages 89–109, Heidelberg. Springer-Verlag.

Lawall, A., Schaller, T., and Reichelt, D. (2014b). Enterprise Architecture: A Formalism for Modeling Organizational Structures in Information Systems. In Barjis, J. and Pergl, R., editors, *Enterprise and Organizational Modeling and Simulation*, volume 191 of *Lecture Notes in Business Information Processing*, pages 77–95. Springer Berlin Heidelberg.

Lawall, A., Schaller, T., and Reichelt, D. (2014c). Local-Global Agent Failover Based on Organizational Models. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 420–427.

Lawall, A., Schaller, T., and Reichelt, D. (2014d). Restricted Relations between Organizations for Cross-Organizational Processes. In *Business Informatics (CBI), 2014 IEEE 16th Conference on*, pages 74–80, Geneva.

Liu, Y. A., Wang, C., Gorbovitski, M., Rothamel, T., Cheng, Y., Zhao, Y., and Zhang, J. (2006). Core Role-based Access Control: Efficient Implementations by Transformations. In *Proceedings of the 2006 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation*, PEPM '06, pages 112–120, New York, NY, USA. ACM.

Sandhu, R. S. (1992). The Typed Access Matrix Model. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, SP '92, pages 122–136, Washington, DC, USA. IEEE Computer Society.

Sandhu, R. S. (1998). Role-Based Access Control. *Advances in Computers*, 46:237–286.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-Based Access Control Models. *Computer*, 29(2):38–47.

Saunders, G., Hitchens, M., and Varadharajan, V. (2001). Role-based Access Control and the Access Control Matrix. *SIGOPS Oper. Syst. Rev.*, 35(4):6–20.

Seufert, S. E. (2002). *Die Zugriffskontrolle*. PhD thesis, Bamberg, Univ., Diss., 2002.

Vahs, D. (2007). *Organisation: Einführung in die Organisationstheorie und -praxis*. Schäffer-Poeschel.

Williamson, G., Sharoni, I., Yip, D., and Spaulding, K. (2009). *Identity Management: A Primer*. Mc Press Series. MC Press Online.