PhD-FSTC-2017-29
The Faculty of Sciences, Technology and Communication

DISSERTATION

Presented on 04/05/2017 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Diego Agustín AMBROSSIO
Born on 20 December 1983 in Ciudad de Buenos Aires (ARGENTINA)

NON-MONOTONIC LOGICS FOR ACCESS CONTROL: DELEGATION REVOCATION AND DISTRIBUTED POLICIES

Dissertation defence committee

Dr. Leon van der Torre, dissertation supervisor
Professor, Université du Luxembourg

Dr. Marcos Cramer
Research Associate, Université du Luxembourg

Dr. Sjouke Mauw, Chairman
Professor, Université du Luxembourg

Dr. Christian Straßer
Professor, Ruhr-University Bochum

Dr. Marc Denecker, Vice Chairman
Professor, KU Leuven

Non-Monotonic Logics for Access Control: Delegation Revocation and Distributed Policies

Diego Agustín Ambrossio

Acknowledgements

"El mundo académico se nutre de la circulación libre de información. Cada uno aporta (literalmente) un granito de arena, y así se hace cada ladrillo. A veces viene un Newton, un Einstein, un Bohr, un Mendel, y trae él solo treinta ladrillos, pero en general es así: granito a granito." *ANÓNIMO*

"The academic world nourishes from the free circulation of information. Each one contributes (literally) one tiny grain of sand, and like this we build each brick. Sometimes, someone like a Newton, an Einstein, a Bohr, a Mendel comes by and brings by himself thirty bricks, but in general it is like this: tiny grain by tiny grain." *ANONYMOUS*

I would like to thank everybody that contributed their grain of sand (or brick) to my life—leading to this moment—so I can leave my tiny grain of sand in the academic world.

Abstract

Access control is concerned with the policies and mechanisms that permit or deny the use of a resource or capability. In a distributed environment, the system's access control policy may be scattered among different principals (e.g. users, processes, nodes, sub-systems, etc.). Many logics have been developed and applied to access control, most of which use a modality *says*. The *says* statements issued may become part of the access control policy. The main advantage of such logics is that they allow to model and reason about common primitives in access control, including *delegation*, *revocation* and *denial* of rights. The revocation and denial of access rights lead to non-monotonic behaviour of the system. We analyze delegation revocation and says-based logics using several formal approaches. First, we develop a logic called Trust Delegation Logic (TDL) to reason about the trust between principals. We equip this logic with two epistemic operators to accommodate for different levels of trust and with a non-monotonic negation by failure. Second, we systematically study the relation between the reasons for revocating (expressed in TDL) and the graph-theoretic definitions of revocation schemes. TDL allows us to formulate a desirable property that a graph-theoretically defined revocation framework should satisfy. Third, we use distributed autoepistemic logic with inductive definitions (dAEL(ID)) to model the behaviour of the *says* operator. dAEL(ID) is a non-monotonic logic, thus allowing to naturally capture the behaviour required for a issuing authorization denials. Finally, we describe a query-based decision procedure for dAEL(ID). We specify a meta-reasoning procedure in the knowledge-based system IDP that allows to examine the local access control policy of each principal and extract which statements have to be queried to other principals.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The access control problem can be described as "*Who* (or *what*) can access *which* (protected) resource". Access control is an important topic in computer security. It consists of methods, mechanisms, procedures and/or processes to determine who has the right to access a particular resource. In general, access control is concerned with the policies and mechanisms that permit or deny the use of a resource or capability. The basic elements of an access control model can be summarized as follows:

- The *access request*: representing the target *resource* (e.g. a file, a communication channel) and the *type of access* requested (e.g. read/write file, open/close channel).
- The *source* of the request: often called *principal*, representing an abstract entity (e.g. user, process, communication channel) which needs to be granted access to interact with a resource.
- The *reference monitor*: in charge of determining whether or not the principal has a access rights, i.e. is authorized to user the resource according to the system's *policy*.

There are several approaches to access control, so called *access control models*. We give a brief summary of some of the most relevant access control models:

- *Attribute-based Access Control (ABAC)*: access rights are granted to principals through the use of policies which evaluate one or more attributes, e.g. principals' attributes, resource attributes, context and environment, etc [HFK+13].
- *Role-Based Access Control (RBAC)*: access rights are pre-defined based on roles that carry specific set of privileges associated to the role and which principals are assigned to each role [FSG+01].
- *Ownership-Based Access Control (OBAC)*: the principal owner of a resource determines who has which access rights to the resource.

In the simplest scenario, the authorization specification for the system is given in the *access control policy*, and the reference monitor has a complete view over the system's policy.

In a distributed environment the access control policy of the system can be scattered among different nodes, sub-systems, etc.

Logic is a suitable general purpose tool to analyze and reason about access control policies. Many logics have been developed and applied to access control, most of which use a modality $k\,says$ indexed by a principal k [Aba08, GP12, Gen12]. $k\,says\,\varphi$ is usually explained informally to mean that k supports φ [Aba08, GP12, Gen12]. $k\,says\,\varphi$ means that k has issued statements that—together with additional information present in the system—imply φ. Most of these logics are designed for application in a system based on *proof-carrying authorization* [AF99, MP11]. Different access control logics vary in their account of which rules of inference and which additional information may be used in deriving statements that k supports from the statements that k has explicitly issued. The statements that different principals can issue may become part of the access control policy.

The main advantage of such logics is that they permit to model and reason about common primitives in access control. The *says* operator can be conveniently used (among other behaviours) to model *delegation of rights* from one principal to another.

The delegation of rights has its complementary operations, those which remove rights e.g. to regain full control of the resource again. On the one hand we can perform a *revocation* of rights that the principal already had been granted. On the other hand we can perform a *denial* of rights to a principal. The difference is that a denial of rights can be issued even *before* the principal has acquired the concerning rights, and it should forbid that other principals effectively grant rights to the affected principal.

Delegation and Revocation. In ownership-based frameworks for access control, it is common to allow principals the possibility of delegating permissions and delegate administrative rights to other principals in the system. Often it is also desirable to grant a principal the right to further grant permissions and administrative rights to other principals. This may lead to delegation chains starting at a source of authority (the principal owner of a resource) and passing on certain permissions to other principals in the chain. Furthermore, such frameworks commonly allow a principal to revoke a permission that she granted to another principal.

Several *revocation schemes* have been discussed in the context of database management systems [GW76, Fag78, BJS96, BSJ97]. Hagström et al. [HJPPW01] have presented a comprehensive framework for classifying possible revocation schemes. The framework's design decisions are carried over from these database management systems and are often not fully motivated.

Depending on the reasons for the revocation, different ways to treat the chain of principals whose permissions depended on the second principal's delegation rights can be desirable. For example, if one is revoking a permission given to an employee because he is moving to another position in the company, it makes sense to keep in place the permissions of principals who received their permissions from this employee; but if one is revoking a permission from a user who has abused his rights and is

hence distrusted by the user who granted the permission, it makes sense to delete the permissions of principals who received their permission from this user. Any algorithm that determines which permissions to keep intact and which permissions to delete when revoking a permission is called a revocation scheme. Revocation schemes are usually defined in a graph-theoretical way on the graph that represents which authorizations between the principals are intact.

We identify a number of problems with Hagström et al.'s framework and the definitions of the revocation schemes. We refine Hagström et al.'s framework and fully motivate our choices based on the *reasons* a principal may have for revocating an authorization.

Non-monotonicity in Access Control. A logic is monotonous if adding new assumptions never leads to less statements being derivable. Monotonicity is a property usually assumed by existing access control logics. Thus, the following property inherently holds: "new statements cannot lead to *less* access rights". Additionally, monotonicity implies that it is impossible to derive a statement of the form $\neg k \mathit{says}\, \varphi$ from the fact that principal k has not issued any statement implying φ. However, being able to derive statements of the form $\neg k \mathit{says}\, \varphi$ makes it possible to model access denials naturally in a *says*-based access control logic: Suppose A is a professor with control over a resource r, B is a PhD student of A who needs access to r, and C is a postdoc of A supervising B. A wants to grant B access to r, but wants to grant C the right to deny B's access to r, for example in case B misuses her rights. A natural way for A to do this using the *says*-modality is to issue the statement $(\neg C \mathit{says}\, \neg \mathit{access}(B, r)) \Rightarrow \mathit{access}(B, r)$. This should have the effect that B has access to r unless explicitly C denies him access.

As explained in Chapter 4, only a non-monotonic *says*-based logic can ensure that access denials work in this way. This motivates our development of a novel *says*-based access control logic that is non-monotonic.

1.2 Research Questions

We have described how logical formalisms are aimed to achieve different interests in access control. We will focus on the development of new logical approaches to address the following research questions.

Research Question 1. *How to formally characterize the different reasons for revoking or denying access rights?*

The literature on revocation schemes usually motivates the distinction between different kids of revocations with reference to varying reasons for revoking. However, the revocation schemes are then defined in an *ad hoc* way with no systematic link to those reasons. This informal treatment may lead to discrepancies between the originally intended behaviour and the resulting revocation schemes proposed. As a remedy to this problem we have developed a logic called Trust Delegation Logic (TDL) to formalize the reasons that a principal may have to revoke or deny an access right.

Research Question 2. *Can the different revocation schemes be defined in such a way that they are in line with the different reasons for revocating?*

It is important to ensure that the reasons a principal has to perform a revocation coincide with the revocation scheme that is to be applied. The reasons a principal may have to revoke a previously granted access right are based on trust, and the revocation schemes refer to a changes in the delegation graph representing the actual situation of the system. Thus, it is crucial that the link between these two is sound and reflects the desired behaviour.

Research Question 3. *How can a says-based access control logic support denials in a natural way?*

To revoke an access right from a principal intrinsically assumes that this access right had been previously granted. When denying a principal from access rights to a resource, this assumption is no longer valid since the denial can be issued even before the principal has been granted the concerning rights. Moreover, a denial should

forbid that other principals effectively grant rights to the affected principal. Denials can cause a non-monotonic behaviour in which granting new access rights does not necessarily result in a principal having more access rights.

Research Question 4. *Can a decision procedure for such an access control logic be developed?*

An important property for any logic is its decidability, that is, the existence of an effective method to determine if a formula is valid or not. This is especially relevant for a logic meant to be implemented as part of a system. The existence of a decision procedure ensures that the logic is decidable.

1.3 Methodology

We use a variety of formal approaches to address the research questions. We analyze delegation revocation and *says*-based logics using techniques from social choice theory, epistemic logic, and logic programming. We now examine in detail the methodology used to approach each research question.

Research Question 1. How to formally characterize the different reasons for revoking or denying access rights?

We develop the logic TDL to reason about the trust between principals. We equip this logic with two epistemic operators to accommodate for different levels of trust. We allow principals to make public announcement statements describing their access control policy.

Research Question 2. Can the different revocation schemes be defined in such a way that they are in line with the different reasons for revocating?

As mentioned in Section 1.1, we identify some problems in other approaches to delegation revocation such as Hagström et al.'s framework. In order to ensure that our refined framework does not itself suffer from similar problems, we systematically study the relation between the reasons for revocating and the graph-theoretic definitions of

revocation schemes. We use TDL in order to formalize reasons for revocating based on trust and distrust.

TDL allows us to formulate a postulate (or axiom) i.e. a desirable property that the revocation schemes must satisfy. This desirable property is based on a correspondence between revocation schemes and reasons for revocating, and requires the revocation schemes to be defined in such a way that access is granted whenever this is justifiable on the basis of the reasons for granting and revocating. This 'axiomatic method' originates from social choice theory, where it was first applied to the study of electoral systems.

Research Question 3. How can a says-based access control logic support denials in a natural way?

We use distributed autoepistemic logic with inductive definitions (dAEL(ID)) [VHCBD16] to model the behaviour of the *says* operator. Autoepistemic logic considers an agent's theory to be a complete characterization of what the agent knows [Moo85a]. Similarly, we assume that the statements issued by a principal is a complete characterization of what the principal supports.

dAEL(ID) is a non-monotonic logic, thus allowing to naturally capture the behaviour required for a issuing authorization denials.

Research Question 4. Can a decision procedure for such an access control logic be developed?

We describe a query-based decision procedure. For this, we use concepts of logic programming and specify a meta-reasoning procedure in the knowledge-based system IDP [PDJD16], that allows to examine the local access control policy and extract which statements have to be queried to other principals.

1.4 Thesis Overview

The rest of this thesis is organized as follows.

- **Chapter 2: Preliminaries.**

  We introduce some basic concepts that will be used in later chapters, especially Chapter 4. These topics include an overview of classical first order logic (FO), three-valued logic, extensions to FO an the IDP framework. We conclude the chapter with basic definitions of lattices and fixpoints, and some important results from approximation fixpoint theory.

- **Chapter 3 : Delegation Revocation.**

  We study the revocation of delegated access rights and propose a framework to reason about the reasons that the principals may have to perform such revocation based on trust.

- **Chapter 4: A Query-Driven Decision-Procedure for Distributed Autoepistemic Logic with Inductive Definitions.**

  We first give a summary of Autoepistemic logic (AEL) and Distributed Autoepistemic logic (dAEL). We motivate its use for access control and define a query-driven decision procedure for the well-founded semantics of dAEL(ID) (distributed autoepistemic logic with inductive definitions).

- **Chapter 5: Abstract Cumulative Aggregation.**

  The scope of this chapter is out of the main topic of this thesis; it is result of parallel work in the domain of deontic logic. Thus, this chapter is self-contained and can be studied in isolation.

  We develop an abstract theory of normative reasoning. We focus on conditional obligations. This abstract framework allows us to reason about the relation among conditional norms from an aggregative point of view. Similar programs had been followed from a transitive point of view. We motivate our choice for the aggregative perspective and briefly contrast similarities and differences with related approaches.

- **Chapter 6: Conclusion and Further Work** We summarise the contributions of this thesis, draw some conclusions, and make suggestions for future work.

1.5 Details of Contributions

The writings in this Ph.D. thesis are composed of the results obtained by the collaboration with colleagues. Thus, in the following a comprehensive summary of individual and collective contributions to this thesis:

- **Chapter 3 : Delegation Revocation.**

  This chapter is based on a joint paper with Marcos Cramer and Pieter Van Hertum [CAV15].

  The formal definition of Trust Delegation Logic (TDL) and Theorem 3.1 were provided by Marcos Cramer.

- **Chapter 4: A Query-Driven Decision-Procedure for Distributed Autoepistemic Logic with Inductive Definitions.**

  This chapter is based on joint work with Marcos Cramer and Pieter Van Hertum [CAV15, AC17].

  The definition of the communication procedure (Algorithms 5 and 6) and the correctness of the communication procedure (Lemmas 4.1, 4.2, 4.3, and Theorem 4.1) were provided by Marcos Cramer.

- **Chapter 5: Abstract Cumulative Aggregation.**

  This chapter is based on a joint paper with Xavier Parent and Leon van der Torre [AXL16].

  The proofs for Theorem 5.2, Proposition 4, Proposition 8 and Proposition 10; and the comparison between our framework with Tossato et al. and Parent and van der Torre frameworks and Theorem 5.3 were provided by Xavier Parent.

Chapter 2

Preliminaries

Abstract. In this chapter we introduce concepts and notions needed and used in later chapters, especially in Chapter 4. Even though most of them are well-known logical formalisms, we provide a brief introduction in order to fix our notation and terminology, and in order to aid the reader throughout the rest of the thesis.

In Section 2.1 we take a look to (classical) first-order logic (FO). Later, in Section 2.2 we introduce three-valued logic. In Section 2.3 we descibe the IDP framework and finally in Section 2.4, we describe some basic notions of lattices, fixpoints and approximation fixpoint theory.

2.1 First-Order Logic

For those readers unfamiliar with FO, we present the full definition of the syntax and semantics of this logic.

FO uses two types of symbols: *logical* and *non-logical* symbols. The logical symbols are fixed and they have a precise meaning in the language, whilst the non-logical symbols are context-dependant, these are declared in the language's *signature* (see Definition 2.1 bellow).

The set of logical symbols of FO contains: *variables* ($x, y, z, \ldots$), *logical connectives* ($\neg, \wedge, \vee, \Rightarrow, =, \ldots$), quantifiers ($\forall, \exists$) and punctuation symbols ('(', ')').

Definition 2.1 (Signature)**.** *A* signature *is a pair* $\Sigma = (\sigma_P, \sigma_f)$ *consisting of non-logical symbols, where* σ_P *and* σ_f *are disjoint sets, called* predicate symbols *and* function symbols *respectively, described as follows:*

(i) Function Symbols*: A (countable, possibly empty) set of function symbols* $f_0, f_1, ...,$ *where each function symbol* f *has an associated* arity ≥ 0 *i.e. the number of arguments of the function.*

(ii) Predicate symbols*: A (countable, possibly empty) set of predicate symbols* $P_0, P_1, ...,$ *where each predicate symbol* P *has an associated* arity ≥ 0 *i.e. the number of arguments of the predicate.*

We often use P/n (f/n) to denote the predicate symbol P (respectively function symbol f) with arity n.

We call function symbols with arity 0 *constants*.

We call predicate symbols with arity 0 *propositional variables*.

In order to define formulas for our language, we first introduce the concept of *term*. These are the basic building blocks from which formulas are built.

Definition 2.2 (FO-terms)**.** *Given a signature* Σ*, the set of* Σ-terms *is inductively defined as follows:*

(i) Every constant *and every* variable *is a* term*.*

(ii) If $t_1, ..., t_n$ *are* terms *and* f *is a function of arity* n*, then* $f(t_1, ..., t_n)$ *is a* term*.*

Definition 2.3 (FO-Syntax)**.** *Given a signature* Σ*, a* formula *of FO is inductively defined as follows:*

(i) $\bot$ *is an* atomic formula*.*

(ii) Every predicate symbol of arity 0 is an atomic formula.

(iii) If t_1 *and* t_2 *are terms, then* $t_1 = t_2$ *is an atomic formula.*

(iv) *If* $t_1, ..., t_n$ *are terms and* P *is a predicate symbol of arity* $n > 0$*, then* $P(t_1...t_n)$ *is an atomic formula.*

(v) *Every atomic formula is a* formula*.*

(vi) *For every two* formulae φ *and* ψ*,* $(\varphi \wedge \psi)$ *and* $\neg\varphi$ *are also formulae.*

(vii) *For any* variable x_i *and any formula* φ*,* $\forall x_i \, \varphi$ *is also a formula.*

The set of Σ-formulae can also be defined by the following EBNF rule:

$$\varphi := \bot \mid P/0 \mid P(t, \ldots, t) \mid \neg\varphi \mid \varphi \wedge \varphi \mid t = t \mid \forall x\varphi$$

We define $\mathscr{L}_{FO}^{\Sigma}$ to be the set of formulae over the signature Σ.

The expresions $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$ and $\exists x\varphi$ are treated as abbreviations for $\neg(\neg\varphi \wedge \neg\psi)$, $\neg\varphi \vee \psi$, $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$, and $\neg\forall\neg\varphi$ respectively. We write $\bar{t}$ for tuples of terms $t_1, \ldots, t_n$. We write $\bar{x}$ for a tuple of variables $x_1, \ldots, x_n$ and the formula $\forall\bar{x}\varphi$ represents the formula $\forall x_1 \forall x_2 \ldots \forall x_n \varphi$.

Definition 2.4 (Sentence)**.** *A* sentence *is a formula without free variables, i.e each variable is associated to a quantifier.*

The subset of the language $\mathscr{L}_{FO}$ that does not use $\forall$ or $=$, and in which all predicates have arity 0 is called *propositional logic*, denoted by $\mathscr{L}_{Prop}$.

Definition 2.5 (FO-Theory)**.** *An FO* theory T *is a set consisting of FO sentences.*

Now that we have defined the syntax of FO, we proceed to define its semantics. A structure represents a state of affairs.

Definition 2.6 (Structure)**.** *A* structure *over a signature* Σ *is a tuple* (D, I)*, where* D *is a non-empty set called the domain of* Σ*, and* $\mathcal{I}$ *is an assignment function that assigns an interpretation to each symbol in* Σ*. For a predicate symbol* P *of arity* n*, the interpretation* $P^{\mathcal{I}}$ *is a subset of* D^n*; for a function symbol* f *of arity* n*,* $f^{\mathcal{I}}$ *is a function from* D^n *to* D*.*

We assume that every $d \in D$ is also a constant symbol with interpretation $d^I = d$.

We call a structure *finite* if its domain is finite.

We now define a valuation for the formulae of FO with respect to a structure S.We use truth values $\mathbf{t}$ for truth, $\mathbf{f}$ for falsity.

Definition 2.7 (FO-Valuation)**.** *We inductively define a valuation of formulas with respect to a structure $S = (D, I)$ as follows:*

$\bot^S = \mathbf{f}$	*for any structure* S
$(P(\bar{t}))^S = \mathbf{t}$	*iff* $t^I \in P^I$
$(t_1 = t_2)^S = \mathbf{t}$	*iff* $t_1^I = t_2^I$
$(\varphi_1 \wedge \varphi_2)^S = \mathbf{t}$	*iff* $(\varphi_1)^S = \mathbf{t}$ *and* $(\varphi_2)^S = \mathbf{t}$
$(\neg\varphi)^S = \mathbf{t}$	*iff* $(\varphi)^S = \mathbf{f}$
$(\forall x\ \varphi)^S = \mathbf{t}$	*iff for each* $d \in D$, $(\varphi[x/d])^S = \mathbf{t}$

The expression $[x/d]$ represents the uniform substitution of all free occurrences of the variable x by the domain element d.

We define the valuation $T^S \in \{\mathbf{t}, \mathbf{f}\}$ of an FO theory T by $T^S = \mathbf{t}$ iff $\varphi^S = \mathbf{t}$ for all $\varphi \in T$.

The notion of a *model* is defined as follows. A model is any structure that makes all formulae in a theory *true*.

Definition 2.8 (Model)**.** *A structure S is a* model *of a theory T iff $T^S = \mathbf{t}$.*

When a formula φ is true with respect to a structure S, i.e. $\varphi^S = \mathbf{t}$, we say that "S satisfies φ", or "S is a model of φ", with notation $S \models \varphi$.

2.2 Three-Valued Logic

Three-valued logic is a particular kind of many-valued logic in which there are three truth values, *truth* (**t**), *falsity* (**f**), and a third value specifying an *undetermined* truth value. We call this third value *undefined* (**u**). We follow what is known as Kleene's "strong logic of indeterminacy" [Kle38] which we also call Kleene's three-valued logic. This logic was first conceived to describe partial recursive relations–that are interpreted as propositional functions of natural numbers–describing partial recursive functions. The correspondence between the relation and the function can take a value of 1 (true), 0 (false), or be undefined. The undefined value can be thought of as neither true nor false.

The logical connectives applied to these relations combine these three truth values as shown below.

The language of three-valued logic $\mathscr{L}_{3val}$ is defined as the language of propositional logic $\mathscr{L}_{Prop}$. The main difference lies on how the logical connectives are semantically defined. We define the semantics of this logic by presenting the value of the truth function for the logical connectives.

Just for clarity's sake, Tables 2.2, 2.3 and 2.4 should be read as follows. The top row represents the truth values assigned to the formula φ. The left-most column represents the truth values assigned to the formula ψ. Inclosed in the remaining central rows/columns are the value assigned resulting from the logical operation.

The truth functions of connectives are defined as follows:

2.3 IDP and FO(ID)

Following [DBBD14a] we use the term FO($\cdot$) for a family of extensions of first-order logic. FO($\cdot$) is developed with the purpose to combine ideas from multiple domains of knowledge representation, logic programming and non-monotonic reasoning in a conceptually clear manner. The basis of this family of languages lies in first order

φ	$\neg\varphi$
t	**f**
f	**t**
u	**u**

Table 2.1: Kleene truth table for $\neg\varphi$

$\varphi \wedge \psi$	**t**	**f**	**u**
t	**t**	**f**	**u**
f	**f**	**f**	**f**
u	**u**	**f**	**u**

Table 2.2: Kleene truth table for $\varphi \wedge \psi$

$\varphi \vee \psi$	**t**	**f**	**u**
t	**t**	**t**	**t**
f	**t**	**f**	**u**
u	**t**	**u**	**u**

Table 2.3: Kleene truth table for $\varphi \vee \psi$

$\varphi \Rightarrow \psi$	**t**	**f**	**u**
t	**t**	**f**	**u**
f	**t**	**t**	**t**
u	**t**	**u**	**u**

Table 2.4: Kleene truth table for $\varphi \Rightarrow \psi$

logic, extended with new language constructs from the fields of logic programming, constraint programming and non-monotonic reasoning. As a trade-off for extending the language, we restrict its semantics to finite domains.

IDP [DWD11] is a Knowledge Base System which supports an FO($\cdot$) language, denoted by FO($\cdot$)$^{\text{IDP}}$. IDP combines a declarative specification (*knowledge base*), written in an extension of first-order logic, with an imperative management of the specification via the Lua [IHC96] scripting language. The extension of first-order logic supported by IDP allows for inductive definitions. Inductive definitions can be (loosely) understood as logic programs in which clause bodies can contain arbitrary first-order formulas. The combination of the declarative specification and the imperative programming environment makes this logic programming tool suitable for solving a large variety of different problems.

IDP supports multiple *inferences* that can be used to perform a range of different reasoning tasks on a given specification. Here we only define two of them, as they are needed in Chapter 4.

The logical system supported by IDP is FO(ID,Agg,PF,T). This logic is an extension of first order logic (FO) with inductive definitions, aggregates, partial functions and types. To ease the presentation and focus only in relevant components

of this rich language, we restrict our attention to th fragment FO(ID) i.e. first order logic with inductive definitions.

First we define some basic notions. These extend those already presented in Section 2.1 for FO logic. For those definitions which remain invariable we refer to the definitions present in Section 2.1.

An FO(ID) specification consists of a *vocabulary*, a *theory* and a possibly *partial* structure. We first modify the definition of an FO signature to what in IDP is called *vocabulary*. The difference here is the addition of *types* or *sorts*.

Definition 2.9 (Vocabulary Σ)**.** *A* vocabulary Σ *is a tuple* $\Sigma = (\sigma_P, \sigma_f, \sigma_T)$ *consisting of non-logical symbols, where* σ_P*,* σ_f*, and* σ_T *are disjoint sets, called* predicate symbols*,* function symbols*, and* type symbols *respectively, described as follows:*

(i) Function Symbols*: A (countable, possibly empty) set of function symbols* $f_0, f_1, ...,$ *where each function symbol* f *has an associated* arity ≥ 0*, and a* type $[\tau_1, \ldots \tau_n] \to \tau_{n+1}$*, which is a tuple of type symbols* $\tau_i \in \sigma_T$*.*

(ii) Predicate symbols*: A (countable, possibly empty) set of predicate symbols* $P_0, P_1, ...,$ *where each predicate symbol* P *has an associated* arity ≥ 0*, and a* type $[\tau_1, \ldots \tau_n]$*, which is a tuple of type symbols* $\tau_i \in \sigma_T$*.*

We write $P(\tau_1, \ldots, \tau_n)$ for a predicate P with type $[\tau_1, \ldots, \tau_n]$ and $f(\tau_1, \ldots, \tau_n) : \tau$ for a function f with type $[\tau_1, \ldots, \tau_n] \to \tau$.

The definitions of terms, formulae and sentences are analogous to Definitions 2.1, 2.3 and 2.4 respectively, including the type restrictions imposed by the typed vocabulary.

An IDP theory consist of *sentences* and *inductive definitions*.

Definition 2.10 (Inductive Definition)**.** *An* inductive definition Δ *is a set of rules of the form:* $\forall \overline{x} : P(\overline{x}) \leftarrow \varphi(\overline{y})$ *where* $\overline{y} \subset \overline{x}$ *and* $\varphi(\overline{y})$ *is a FO formula. We call*

predicate P the head *of the rule, and a* defined predicate *of the definition. Any other predicate or function symbol in Δ is called a parameter.*

Because of their rule-based nature, formal inductive definitions also bear strong similarities in syntax and formal semantics with logic programs. A formal inductive definition could also be understood intuitively as a logic program which has arbitrary formulas in the body and which defines only a subset of the predicates in terms of parameter predicates not defined in the definition.

Definition 2.11 (Theory)**.** *A* theory $\mathcal{T}$ *over a vocabulary Σ is a set that consists of* sentences *and* inductive definitions *with symbols in Σ.*

This concludes the syntactical account. We now proceed to define the semantical notions of FO(ID). We first define the notion of *partial set*, which is the three-valued analogue of the two-valued notion of a set.

Definition 2.12 (Partial Set)**.** *A* partial set *on the domain D is a function from D to $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, where $\mathbf{t}$, $\mathbf{f}$ and $\mathbf{u}$ stand for the three truth-values* true, false *and* undefined.

A partial set is two-valued (or total) if $\mathbf{u}$ does not belong to its range.

We extend the concept of structure as presented in Definition 2.6 to the concept of a *partial structure* in order to accommodate for partial sets and types. Given a vocabulary Σ, a partial structure gives an interpretation to the elements of Σ.

Definition 2.13 (Partial Structure)**.** *A* partial structure *over a fixed vocabulary $\Sigma = (\sigma_P, \sigma_f, \sigma_T)$ is a tuple $(D_{\tau_1}, \ldots, D_{\tau_n}, I)$, where $\sigma_T = \{\tau_1, \ldots, \tau_n\}$ and for each D_{τ_i} is a non-empty set, called the domain of type τ_i, and $\mathcal{I}$ is an assignment function that assigns an interpretation to each symbol in Σ. For a predicate symbol $P[\tau_1, \ldots, \tau_n]$ of arity n, the interpretation $P^{\mathcal{I}}$ is a partial subset of $D_{\tau_1} \times \ldots \times D_{\tau_n}$; for a function symbol $f[\tau_1, \ldots, \tau_n] \to \tau$ of arity n, $f^{\mathcal{I}}$ is a partial function from $D_{\tau_1} \times \ldots \times D_{\tau_n}$ to D_τ.*

When the predicate symbol P has arity 0, i.e. is a propositional variable, $P^{\mathcal{I}}$ is just an element of $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$.

We call a partial structure $S = (D_\tau, \mathcal{I})$ *finite* if and only if its domain D_τ is finite (i.e. the domain D_τ is finite for every $\tau \in \sigma_\tau$). We call a partial structure *two-valued* (or *total*) iff $P^{\mathcal{I}}(\bar{d}) \in \{\mathbf{t}, \mathbf{f}\}$ for all $P \in \sigma_P, \bar{d} \in D_{\tau_1} \times \ldots \times D_{\tau_n}$, and f^I is a total function for all $f \in \sigma_f$.

A precision order can be defined on partial structures.

Definition 2.14 (Ordering-Structure)**.** *Given two partial structures $S = (D, \mathcal{I})$ and $S' = (D, \mathcal{I}')$, we write $S \leq_p S'$ (and say S'* is more precise than *S, or S'* expands *S) iff for every function symbol f of arity n and every tuple $(\bar{d}, d) \in f$ of domain elements such that $f^{\mathcal{I}}(\bar{d}) = d$, we have $f^{\mathcal{I}'}(\bar{d}) = f^{\mathcal{I}}(\bar{d})$, and for every predicate symbol P of arity n and every tuple $\bar{d} \in D^n$ of domain elements such that $P^{\mathcal{I}}(\bar{d}) \neq \mathbf{u}$, we have $P^{\mathcal{I}'}(\bar{d}) = P^{\mathcal{I}}(\bar{d})$.*

2.4 Lattices, Operators and Approximation Fixpoint Theory

In Chapter 4, we will make use of lattice theory and a lattice-theoretical method called Approximation Fixpoint Theory (AFT) [DMT00] to define multiple semantics of autoepistemic logic. We recall the basic concepts of lattice theory and some important results from AFT.

A *complete lattice* is a partially ordered set $\langle L, \leq \rangle$ such that every subset $S \in L$ has a *least upper bound*, written $lub(S)$, and a *greatest lower bound*, written $glb(S)$. A complete lattice has a *least element* denoted by $\bot$ and a *greatest element* denoted by $\top$.

An *operator* on a lattice $\langle L, \leq \rangle$ is any mapping $O : L \to L$. An operator O is *monotone* if for every pair (x, y) of elements such that $x \leq y$, we have $O(x) \leq O(y)$.

The main technical result in fixpoint theory is the so called Knaster-Tarski Theorem [Tar55].[a]

[a] This theorem is also known as the Tarski's fixed point theorem.

Theorem 2.1. *Let O be a monotone operator on a complete lattice $\langle L, \leq \rangle$. Then O has a fixpoint and the set of all fixpoints of O is also a complete lattice.*

Using Theorem 2.1 we can ensure that a monotone operator O in a complete lattice has a least fixpoint, written $lfp(O)$. This fixpoint can be calculated as the least upper bound of the of the transfinite sequence O^α given by:

1- $O^0(x) = \bot$

2- $O^{\alpha+1}(x) = O(O^\alpha(x))$, for any ordinal α.

3- $O^\lambda(x) = lub\{O^\alpha(x) \mid \alpha \leq \lambda\}$, with λ a limit ordinal.

Later we will restrict ourselves to use finite domains. In that case, the sequence will converge after finitely many steps, so the third clause in the definition of O^α is not required. Moreover in this case, we can devise an algorithm to calculate the sequence of O^α up to the fixpoint.

After this basic introduction to lattices and operators over lattices we define the basic notions of Approximation Fixpoint Theory.

Given a lattice L, we generate a bi-lattice L^2 with its usual projections: $(x, y)_1 = x$ (first component) and $(x, y)_2 = y$ (second component). Each pair $(x, y) \in L^2$ is used to approximate an element z in the interval $[x, y] = \{z \mid x \leq z \leq y\}$. Not every pair $(x, y) \in L^2$ can be used to approximate elements of L; it is necessary that $x \leq y$. We call these pairs $(x, y) \in L^2$ *consistent* if $[x, y]$ is non-empty, in other words if $x \leq y$. Otherwise, we call these pairs *inconsistent*. We call the set of all consistent pairs L^c. The pairs $(x, x) \in L^2$ are called *exact*. We identify each point $x \in L$ with the exact bi-lattice point $(x, x) \in L^c$.

The *precision ordering* over L^2 is defined as follows: $(x, y) \leq_p (u, v)$ if $x \leq u$ and $v \leq y$. So, $(x, y) \leq_p (u, v)$ holds if and only if the interval $[u, v]$ given by the pair (u, v) is included in the interval $[x, y]$ given by the pair (x, y). Thus, (u, v) is more precise than (x, y).

Figure 2.1 depicts the relation between the intervals $[x, y]$ and $[u, v]$ when $(x, y) \leq_p (u, v)$.

Figure 2.1: Inclusion relation among intervals

If L is a complete lattice, then $\langle L^2, \leq_p \rangle$ is also a complete lattice [Gin88]. But, the collection of consistent pairs does not generally form a complete sub-lattice of L^2.

The main purpose of AFT is to study fixpoints of lattice operators $O : L \to L$ through associated operators in the bi-lattice $A : L^2 \to L^2$ approximating O.

Definition 2.15 (Approximator)**.** *Let $O : L \to L$ be an operator over a lattice L. A lattice operator $A : L^2 \to L^2$ is an* approximator *of O if:*

- *A is $\leq_p$-monotone.*
- *$\forall x \in L : O(x) \in [x', y']$, where $(x', y') = A(x, x)$.*

Approximators map elements of L^c into L^c. We will restrict ourselves to symmetric approximators. An approximators A is symmetric if for all x and y, $A(x, y)_1 = A(y, x)_2$. In the case of symmetric approximators all the fixpoints of interest later to be used to define the semantics of autoepistemic logic in Chapter 4, namely supported, stable, well-founded are uniquely determined by an approximator's restriction to L^c [DMT04, Theorem 4.5]. Thus, when defining an operator, we will only define its restriction to L^c.

Chapter 3

Delegation Revocation

Abstract. In this chapter we introduce *Trust Delegation Logic* (*TDL*), proposed to formalize the different reasons a principal may have for performing a revocation. Hagström et al. [HJPPW01] proposed a framework for classifying revocation schemes, motivated by presenting various scenarios in which the principals have different reasons for revocating. The new logic arises from the observation that there are some problems with Hagström et al.'s definitions of the revocation schemes, which have led us to propose a refined framework. In order to formally study the merits and demerits of desirable definitions of revocation schemes, we propose to apply the axiomatic method originating in social choice theory to revocation schemes. We employ TDL for formulating an axiom, i.e. a desirable property of revocation frameworks. We show that our refined framework, unlike Hagström et al.'s original definitions, satisfy the desirable property that can be formulated using TDL.

3.1 Introduction

In ownership-based frameworks for access control, it is common to allow principals (users or processes) to grant both permissions and administrative rights to other principals in the system. Often it is desirable to grant a principal the right to further grant permissions and administrative rights to other principals. This may lead to

delegation chains starting at a *source of authority* (the owner of a resource) and passing on certain permissions to other principals in the chain.

Furthermore, such frameworks commonly allow a principal to revoke a permission that she granted to another principal. Depending on the reasons for the revocation, different ways to treat the chain of principals whose permissions depended on the second principal's delegation rights can be desirable. For example, if one is revoking a permission given to an employee because he is moving to another position in the company, it makes sense to keep in place the permissions of principals who received their permissions from this employee; but if one is revoking a permission from a user who has abused his rights and is hence distrusted by the user who granted the permission, it makes sense to delete the permissions of principals who received their permission from this user. Any algorithm that determines which permissions to keep intact and which permissions to delete when revoking a permission is called a *revocation scheme*. Revocation schemes are usually defined in a graph-theoretical way on the graph that represents which authorizations between the principals are intact.

Hagström et al. [HJPPW01] have presented a framework for classifying possible revocation schemes along three different dimensions: the extent of the revocation to other grantees (propagation), the effect on other grants to the same grantee (dominance), and the permanence of the negation of rights (resilience). Since there are two options along each dimension, there are in total eight different revocation schemes in Hagström et al.'s framework. This classification was based on revocation schemes that had been implemented in database management systems [GW76, Fag78, BJS96, BSJ97]. The framework's design decisions are carried over from these database management systems and are often not fully motivated. Furthermore, the behaviour of the revocation schemes is dependent on the conflict resolution policy of the system, which is not integrated into the framework.

We identify a number of problems with Hagström et al.'s framework and the definitions of the revocation schemes included in the framework. This motivates

our refined framework, in which the conflict resolution policy is integrated into the framework, and in which the graph-theoretic definitions of the revocation schemes have been modified.

In order to avoid that our refined framework turns out to have undesirable properties like those we identified in Hagström et al.'s framework, we propose to formally study the merits and demerits of various definitions of revocation schemes using the axiomatic method originating in social choice theory. Which behaviour is desirable for a revocation scheme depends on the reasons for performing the revocation. So in order to formulate an axiom, i.e. a desirable property of revocation schemes, we propose a logic, *Trust Delegation Logic* (*TDL*), with which one can formalize the different reasons an agent may have for performing a revocation. We show that our modified graph-theoretic definitions of the revocation schemes, unlike Hagström et al.'s original definitions, satisfy the desirable property that can be formulated using TDL.

The reminder of the chapter is structured as follows. In Section 3.2, we discuss related work, giving an overview of Hagström et al.'s framework as well as of the conflict resolution policies proposed in the literature. In Section 3.3 we motivate and define a refinement to Hagström et al.'s framework. In Section 3.4 we discuss a formal difficulty caused by a simultaneous induction presented in Section 3.3. In Section 3.5, we consider the diverse reasons for revocating on some example scenarios, and sketch how these reasons can be used to formulate the desirable behaviour of the revocation schemes. In Section 3.6, we motivate and define Trust Delegation Logic (TDL). In Section 3.7 we illustrate how the scenarios discussed in Section 3.5 can be formalized in TDL. In Section 3.8 we use TDL to formally formulate a desirable property for revocation frameworks, which our revocation framework satisfies. We summarize and conclude the chapter in Section 3.9.

3.2 Related work

Hagström et al. [HJPPW01] have introduced three dimensions according to which revocation schemes can be classified. These are called *propagation*, *dominance* and *resilience*:

Propagation. The decision of a principal i to revoke an authorization previously granted to a principal j may either be intended to affect only the direct recipient j or to propagate and affect all the other users in turn authorized by j. In the first case, we say that the revocation is *local*, in the second case that it is *global*.

Dominance. This dimension deals with the case when a principal losing a permission in a revocation still has permissions from other grantors. If these other grantors' revocation rights are dependent on the revoker, the revoker can dominate over these grantors and revoke the permissions from them. This is called a *strong* revocation. The revoker can also choose to make a *weak* revocation, where permissions from other grantors to a principal losing a permission are kept.

Resilience. This dimension distinguishes revocation by removal (deletion) of positive authorizations from revocation by issuing a negative authorization which just inactivates positive authorizations. In the first case another principal may grant a similar authorization to the one that had been revoked, so the effect of the revocation does not persist in time. In the second case a negative authorization will overrule any (new) positive permission given to the same principal, so its effect will remain until the negative permission is revoked. We call a revocation of the first kind a *delete* or *non-resilient* revocation, and a revocation of the second kind a *negative* or *resilient* revocation.

Since there are two possible choices along each dimension, Hagström et al.'s framework allows for eight different revocation schemes.

Delegation frameworks that allow issuing negative authorization can bring about a state in which a conflict may arise. If a principal is granted both a positive and a negative authorization for the same object, then we say that these two authorizations

conflict each other. A system's *conflict resolution policy* determines how to resolve such a conflict. Here is a list of possible conflict resolution policies as described by Ruan and Varadharajan [RV02]:

Negative-takes-precedence: If there is a conflict occurring on the authorization for some object, the negative authorizations will take precedence over the positive one.

Positive-takes-precedence: Positive authorizations from i to j take precedence over negative authorizations from k to j for all $k \neq i$. This means that a negative authorization from i to j directly inactivates only positive authorizations from i to j, and leaves other permission to j active.

Strong-and-Weak: Authorizations are categorized in two types, strong and weak. The strong authorizations always take precedence over the weak ones. Conflicts among strong authorizations are not allowed. In conflicts between weak authorizations negative ones take precedence. Note that the intended meaning of *strong* and *weak* in this policy differs from their meaning in Hagström et al.'s dominance dimension.

Time-takes-precedence: New authorizations take precedence over previously existing ones. Note that this policy will make negative authorizations non-resilient.

Predecessor-takes-precedence: If the principal i delegates (possibly transitively) some right to principal j, then authorizations issued by i to some other principal k concerning that right will take precedence over authorizations issued by j to k. In other words, the priority of subjects decreases as the privilege is delegated forward.

Hagström et al. assume the system to have either a negative-takes-precedence or a positive-takes-precedence conflict resolution policy. Note that under a negative-takes-precedence policy, a negative revocations on principal k dominates all positive authorizations to k, so that the difference between weak and strong negative revocations disappears.

3.3 Refining the revocation framework

In this section we first analyze some problems with the revocation framework by Hagström et al. [HJPPW01]. While analyzing the problems, we already informally sketch how we propose to solve them. Next we define a refined revocation framework in which all of these problems have been solved.

3.3.1 Problems with Hagström et al.'s framework

(1) In Hagström et al.'s framework, strong global revocations will propagate forward dominating over all the existing delegation chains, making them even stronger than desired. We illustrated this by an example:

Example 3.1. *User A issues an authorization to users B and C. B also grants this authorization to C. If a strong global delete revocation (in Hagström et al.'s sense) is performed over the authorization from A to B, then the authorization A granted to C is also deleted. But since A granted this authorization to C independently from B, it seems unjustified to delete it (Hagström et al. give no motivation for this behaviour).*

(2) In Hagström et al.'s framework, the choice of a conflict resolution policy is not incorporated into the revocation framwork, even though it affects the behaviour of the dominance dimension. We extend the dominance dimension to incorporate the choice of how to resolve conflicts between positive and negative authorizations in the revocation framework. In our refined framework, there are three choices along the dominance dimension:

- **weak:** The principal performing the revocation only dominates over direct authorizations granted by herself, authorizations from other grantors are kept intact.
- **predecessor-takes-precedence (p-t-p):** The principal performing the revocation dominates over other grantors' authorizations that are dependent on her.

- **strong:** The principal performing the revocation dominates over all other grantors' authorizations.

Note that we now use the terminology from conflict-resolution policies as presented in [RV02] and in Section 3.2 for the choices on the dominance dimension. Hence "strong" now has a different meaning than in Hagström et al.'s framework: As long as Hagström et al.'s framework is combined with a positive-takes-precedence policy, the strong revocations in their framework have the same force as our p-t-p revocations. The strength of our strong revocations can only be achieved in Hagström et al.'s framework by combining it with a negative-takes-precedence policy.

It is not desirable to allow all users who have a delegation right to perform strong revocations. Hence we include in our framework the possibility for a principal to grant to another principal a special right to perform strong revocations to other users.

(3) In Hagström et al.'s framework, delete revocations are supposed to be non-resilient, which according to Hagström et al. means that "another user may issue the same permission that was just revoked, and the effect of the revocation disappears". This property fails to be satisfied in global deletion revocations, as illustrated by this example:

Example 3.2. *User A issues an authorization to user B, and B further grant this authorization to C. If A deletes the authorization given to B, then the authorization from B to C is also deleted. Reissuing the authorization from A to B will not re-instate the authorization from B to C as before the revocation.*

To avoid this problem in our framework, when a delete is performed, we do not delete the forward chain, but just inactivate it.

(4) Hagström et al. motivate the distinction between delete and negative revocations mainly through the notion of resilience as defined in Section 3.2. However, in weak revocations there can be no difference between a resilient and a non-resilient revocation, since a weak revocation does not affect authorizations issued by others than the revoker. They motivate the usage of weak negatives by pointing out that

they are useful for temporary revocations. But since in our framework the forward chain does not get deleted in a delete revocation (see point (3)), a delete can also be easily undone, so that a delete revocation is a sensible choice even when the revocation is likely to be only temporary. Hence we do not need weak negative revocations.

Furthermore, p-t-p and strong deletes would have undesirable effects, as illustrated by the following example:

Example 3.3. *User A issues an authorization to user B, and gives user C the right to perform strong revocations. User C performs a strong delete on B, removing without traces the authorization provided to B by A. Later A realizes that C cannot be trusted to perform strong revocations, and takes away C's right to do so. Even though C can no longer perform strong revocations, the effect of his strong delete remain: B does not have the right originally issued to him by A until someone issues a new authorization to him.*

Hence we do not have a p-t-p or strong delete revocation in our framework, but instead have the distinction between a resilient and a non-resilient negative for p-t-p and strong revocations.

To conclude, if the dominance of a revocation is p-t-p or strong, there are two options along the resilience dimension, non-resilient and resilient. But if the dominance is weak, there is no choice along the resilience dimension, and the revocation is characterized as a "weak delete". So there are five possible choices to be made along the dominance and resilience dimensions: weak delete, p-t-p non-resilient, p-t-p resilient, strong non-resilient, and strong resilient.

(5) Hagström et al. do not allow negative authorizations to be inactivated. The reason they give is that they "do not want a revocation to result in a subject having more permissions than before the revocation". However, the deletion of negative authorizations is allowed, even though it may have the same effect. We do allow negative authorizations to be inactivated, but the only kind of revocation that can

result in a subject having more permissions than before is a revocation of someone's right to perform strong revocations, and in this case this is a desirable property.

3.3.2 The refined framework

Let $\mathbf{S}$ be the set of principals (subjects) in the system, let $\mathbf{O}$ be the set of objects in the system and let $\mathbf{A}$ be the set of access types. For every object $o \in \mathbf{O}$, there is a *source of authority* (SOA), i.e. the manager of object o.

For any $\alpha \in \mathbf{A}$ and $o \in \mathbf{O}$, the SOA of o can grant the right to access α on object o to other principals in the system. Secondly, the SOA can delegate this granting right further. Thirdly, the SOA can grant the right to perform strong revocations and to delegate this right further. Accordingly we have three *permissions*: *access right* (A), *delegation right* (D) and *strong revocation right* (S). We assume that delegation right implies access right. The set $\{A, D, S\}$ of permissions is denoted by $\mathbf{P}$.

Additionally to positive authorizations ($+$), our framework admits four different types of negative authorizations, *p-t-p resilient negative* ($-_{\mathrm{PR}}$), *p-t-p non-resilient negative* ($-_{\mathrm{PN}}$), *strong resilient negative* ($-_{\mathrm{SR}}$) and *strong non-resilient negative* ($-_{\mathrm{SN}}$). The set $\{+, -_{\mathrm{PR}}, -_{\mathrm{PN}}, -_{\mathrm{SR}}, -_{\mathrm{SN}}\}$ of authorization types is denoted by $\mathbf{T}$.

Definition 3.1. *An* authorization *is a tuple* $(i, j, \alpha, o, \tau, \pi, t)$*, where* $i, j \in \boldsymbol{S}$*,* $\alpha \in \boldsymbol{A}$*,* $o \in \boldsymbol{O}$*,* $\tau \in \boldsymbol{T}$*,* $\pi \in \boldsymbol{P}$*,* $t \in \mathbb{Z}$*.*

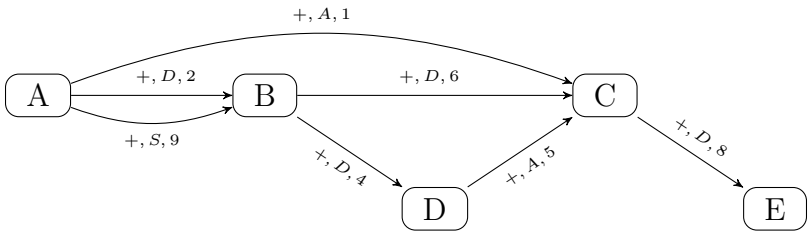
The meaning of an authorization $(i, j, \alpha, o, \tau, \pi, t)$ is that at time point t principal i has granted to principal j an authorization of type τ for permission π concerning access type α on object o. We assume that all authorizations in the system are stored in an *authorization specification*. There is no interaction between the rights of principals concerning different access-object pairs (α, o). For this reason, we can consider α and o to be fixed for the rest of the chapter, and can simplify $(i, j, \alpha, o, \tau, \pi, t)$ to (i, j, τ, π, t).

Since delegation right implies access right, an authorization $(i, j, +, D, t)$ can only be issued if an authorization $(i, j, +, A, t)$ is also issued. By taking the contrapositive,

the connection is reversed for negative authorizations: For $\tau \neq +$, an authorization (i, j, τ, A, t) can only be issued if an authorization (i, j, τ, D, t) is issued.

We visualize an authorization specification by a labelled directed graph, as in Example 3.4, in which A is the SOA. For every authorization (i, j, τ, π, t) in the authorization specification, this graph contains an edge from i to j labelled τ, π, t. We refrain from showing the authorizations that can be implied to exist by the rules specified in the previous paragraph.

Example 3.4. *An authorization specification.*



A negative authorization can inactivate other authorizations in the authorization specification. Which authorizations get inactivated by a negative authorization depends on which type of negative authorization it is. There are three basic ideas governing the inactivation of authorizations: Firstly, non-resilient authorizations can only inactivate previously issued authorizations, whereas resilient authorizations can also inactivate authorizations issued after the negative authorization. Secondly, a strong negative authorization from i to j inactivates every positive authorization from some principal k to j, whereas a p-t-p authorization from i to j only inactivates an authorizations from k to j if k is dependent on i. Thirdly, any authorization that is no longer connected back to the SOA through active authorizations is inactivated.

In order to formally specify which authorizations get inactivated when issuing a negative authorization, we simultaneuosly define the notions of an authorization being *active* and an authorization being *directly inactivated* in Definitions 3.2 and 3.3. The auxiliary notion of a directly inactivated authorization captures the idea

of an authorization from k to j being inactivated by a strong negative authorization from i to j.

Definition 3.2. *An authorization* (i, j, τ, π, t) *is* active *if it is not* directly inactivated *and there are principals* $p_1, \ldots, p_n, p_{n+1}$ *and integers* $t_1, \ldots, t_n$ *satisfying the following properties:*

(i) $p_1 = SOA$, $p_n = i$, $p_{n+1} = j$ *and* $t_n = t$;

(ii) *for* $1 \leq l < n$ *there is an authorization* $(p_l, p_{l+1}, +, \pi', t_l)$ *that is not* directly inactivated, *where* $\pi' = S$ *if either* $\tau \in \{-_{\mathrm{SR}}, -_{\mathrm{SN}}\}$ *or* $\pi = S$, *and* $\pi' = D$ *otherwise;*

(iii) *there do not exist* l, m *with* $1 \leq l \leq m \leq n$ *and an authorization* $(p_l, p_{m+1}, \tau', \pi'', t')$ *such that* $\pi'' = \pi$ *and* $\tau = +$ *if* $m = n$, *and* $\pi'' = \pi'$ *otherwise, and such that either* $\tau' = -_{\mathrm{PN}}$ *and* $t' > t_m$ *or* $\tau' = -_{\mathrm{PR}}$.

Definition 3.3. *An authorization* $(i, j, +, \pi, t)$ *is* directly inactivated *if there is an* active *authorization* $(k_1, j, -_{\mathrm{SR}}, \pi, t_1)$ *or there is an* active *authorization* $(k_2, j, -_{\mathrm{SN}}, \pi, t_2)$ *with* $t_2 > t$.

Definitions 3.2 and 3.3 inductively depend on each other. They should be read as an inductive definition with the well-founded semantics [Den98]. In Section 3.4 we discuss some of the issues resulting from the inductive interdependence of these definitions and later in Chapter 4 we give a full account of inductive definitions and the well-founded semantics.

A principal j has the right to access of type α on object o iff j is the SOA or there is an active authorization of the form $(i, j, \alpha, o, +, A, t)$ or $(i, j, \alpha, o, +, D, t)$. j has the right to perform strong revocations concerning action α on object o iff j is the SOA or there is an active authorization of the form $(i, j, \alpha, o, +, S, t)$. Strong negative authorizations towards the SOA are disallowed.

In Definition 3.4, we define the ten revocation schemes of our refined framework. We use W, P, S, L, G, N, R and D as abbreviations for *weak*, *p-t-p (predecessor-takes-precedence)*, *strong*, *local*, *global*, *non-resilient*, *resilient* and *delete* respectively.

Note that when defining the revocation schemes, we do not need to specify which of the authorizations get inactivated, because Definition 3.2 already tells us what to inactivate. Hence we just specify which authorizations get added and/or deleted.

Definition 3.4. *Let i, j be principals, and $\pi \in \{A, D, S\}$.*

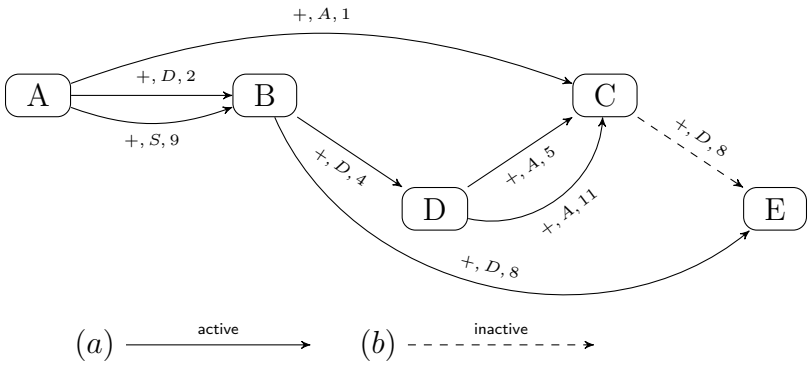
- *A* WGD revocation for permission π from i to j at time t *consists of deleting any authorization of the form* $(i, j, +, \pi, t')$.
- *For $\delta \in$* {P,S} *and $\rho \in$* {N,R}*, a* δGρ revocation for permission π from i to j at time t *consists of issuing the negative authorization* $(i, j, -_{\delta\rho}, \pi, t)$.
- *For $(\delta, \rho) \in$* {(W,D), (P,N), (P,R), (S,N), (S,R)}*, a* δLρ revocation for permission π from i to j at time t *consists of deleting and adding the same revocations as in a* δGρ *revocation from i to j, and – if π is D or S – additionally adding an authorization (i, l, τ, π', t') for every authorization (j, l, τ, π', t') such that $\pi' = S$ if $\pi = S$, and π' is either D or A if $\pi = D$.*

Since delegation right implies access right, a revocation for permission A can only be performed if the corresponding revocation for permission D is performed at the same time.

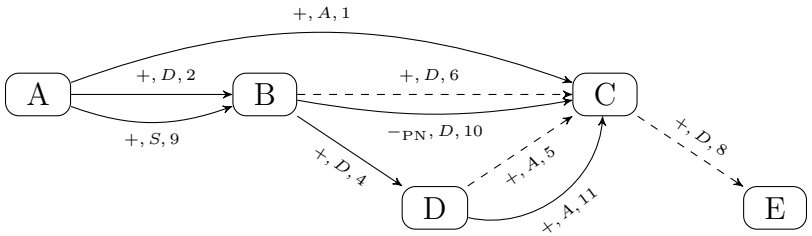
3.3.3 Examples of revocations

Here are ten examples for different revocations from B to C on the authorization specification from Example 3.4. In all examples, we show the effect of simultaneous revocations for permissions A and D. In order to illustrate better the difference between resilient and non-resilient revocations, we show the state of the authorization specification after D reissues the previously issued authorization to C after the revocation.

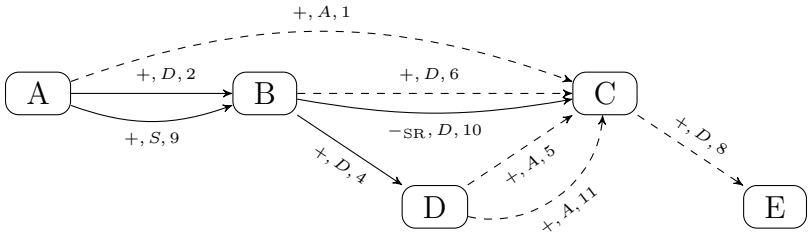
Example 3.5. *Weak Local Delete revocation from* B *to* C



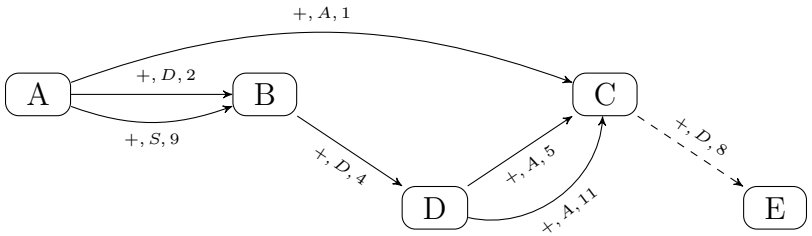
Example 3.6. *P-t-p Global Non-resilient revocation from* B *to* C



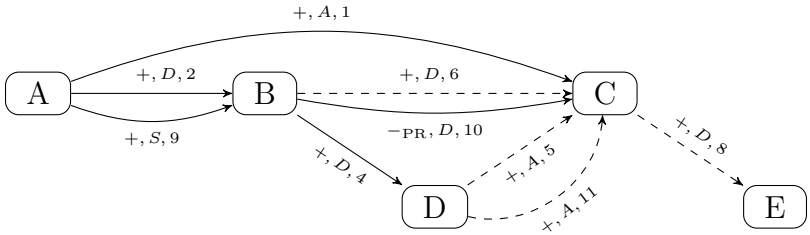
Example 3.7. *Strong Global Resilient revocation from* B *to* C



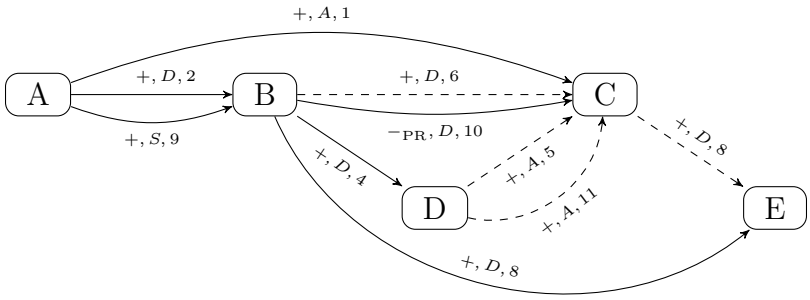
Example 3.8. *Weak Global Delete revocation from* B *to* C



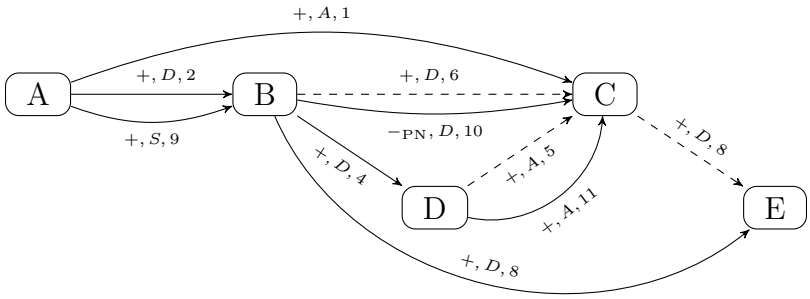
Example 3.9. *P-t-p Global Resilient revocation from* B *to* C



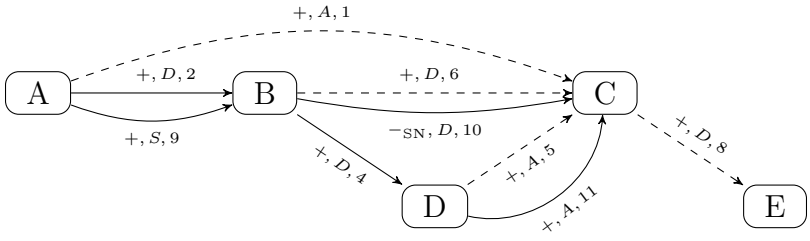
Example 3.10. *P-t-p Local Resilient revocation from* B *to* C



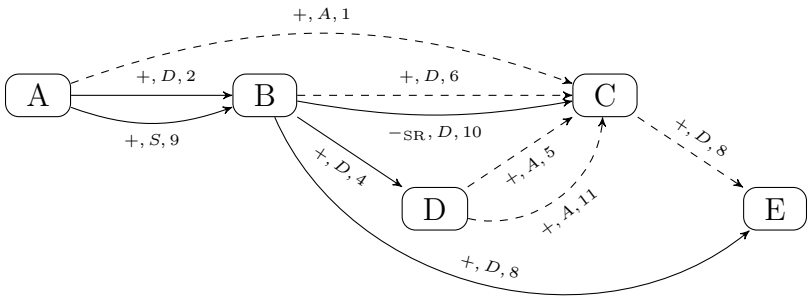
Example 3.11. *P-t-p Local Non-resilient revocation from* B *to* C



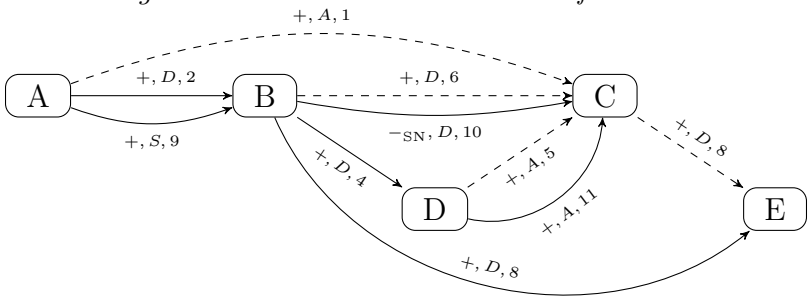
Example 3.12. *Strong Global Non-resilient revocation from* B *to* C



Example 3.13. *Strong Local Resilient revocation from* B *to* C



Example 3.14. *Strong Local Non-resilient revocation from* B *to* C



3.4 Inductive interdependence of Definitions 3.2 and 3.3

Definitions 3.2 and 3.3, in which we define the notions of an authorization being *active* and being *directly inactivated*, depend one on another. This interdependence looks similar to a problematic circular definition, but can actually be understood as an inductive definition, which can be formally interpreted using the well-founded semantics [Den98]. In this section we briefly sketch the theory of inductive definitions under the well-founded semantics, and comment about what practical consequences follow from using an inductive definition for defining the notion of an *active* authorization.

An *inductive definition* of predicates $P_1, \ldots, P_n$ consists of rules of the form $\forall \bar{x}\ P(\bar{x}) \leftarrow \psi$, where $P \in \{P^1, \ldots, P^n\}$ and ψ may itself refer to predicates in $\{P^1, \ldots, P^n\}$. In such a rule, $P(\bar{x})$ is called the *head* of the rule, and ψ the *body* of the rule.

An inductive definition may have a form, which makes it impossible to interpret the defined predicates in a way that is consistent with the rules. For example, the inductive definition $\{P \leftarrow \neg P\}$, which defined P to be the case if $\neg P$ is the case is problematic: If we assume P is false, then $\neg P$ is true, so by the single rule of this definition, P should be true. So P cannot be false. However, since P is defined through this inductive definition, it can only be true if some rule in the inductive definition is true. But the only rule in this definition cannot make P true.

One approach to avoid such problems is to define some syntactic restrictions on the set of rules that ensure that such problems cannot arise. However, such syntactic restrictions are usually too restrictive, barring some inductive definitions that we can intuitively and formally make sense of.

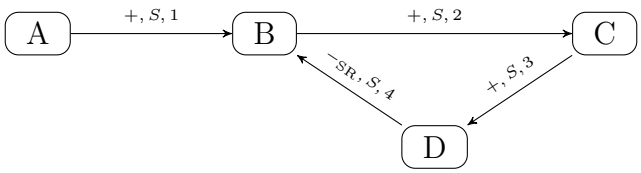
Another approach to avoid such problems is that of using the *well-founded semantics* for inductive definitions, which is a partial semantics, i.e. for problematic definitions like the above, it does not define a truth function for the predicates that were purported to be defined by the inductive definition. But whenever an inductively defined predicate has an intuitively meaningful interpretation, the well-founded semantics formally assigns this interpretation to the predicate [Den98].

The well-founded semantics is defined through an inductive process which involves adding new information about the defined predicates at each step of the induction based on the rules in the inductive definition: This information consists of an assignment of truth values to *domain atoms*; a *domain atom* is a pair $(P, \bar{a})$ – usually written as $P(\bar{a})$ – where P is an n-ary predicate symbol defined in the inductive definition, and $\bar{a} \in D^n$, where D is the domain of the structure. At the first step of the induction, no truth values are assigned to domain conditions. At every subsequent step, the truth value $\mathbf{t}$ may be assigned to a domain atom $P(\bar{a})$ if applying the rules

for P to the *previous* assignment of truth values to domain atoms establishes that $P(\bar{a})$ is true; the truth value **f** may be assigned to domain atoms $P_1(\bar{a_1}), \ldots, P_k(\bar{a_k})$ if applying the rules for $P_1, \ldots, P_k$ to the *resulting* assignment of truth values to domain atoms establishes that $P_1(\bar{a_1}), \ldots, P_k(\bar{a_k})$ are false. This different treatment of the two truth values reflects the inductive nature of the definition, according to which a domain atom can only be true if some rule makes it true, whereas a domain atom should be considered false in the absense of a rule making it true. The *well-founded model* of an inducive definition is defined to be the limit assignment of truth values to domain atoms in such an induction [Den98]. If the well-founded model does not assign either **t** or **f** to every domain atom of the defined predicates, it is a partial model.

The inductive definition of authorizations being *active* and *directly inactivated* can in some contexts lead to a partial well-founded model, i.e. to the notions of *active* and *directly inactivated* not having a coherently determinable meaning. Consider for example the following authorization specification:

Example 3.15.



If we assume that the negative authorization from D to B is active, it directly inactivates the authorization from A to B. But then there is no active chain of authorizations that supports the authorization from D to B, so it would have to be inactive. If on the other hand we assume that the authorization from D to B is inactive, then the authorization from A to B is not directly inactivated, an the chain of authorization from A via B and C to D is active, thus ensuring that the authorization from D to to B is active. So either way we run into a contradiction.

This means that the interpretation of "active" under the well-founded semantics is partial in the context of this authorization specification.

In this example, the problem was caused by principal D using its power to perform strong revocations in order to remove the rights from B, even though D's right to perform strong revocations depended on a delegation chain including B. The problem can be avoided by adding a constraint to the system that disallows principals from using their strong revocation right to remove the strong revocation right from a principal on whom they depend for their strong revocation right.

This constraint needs to be formulated in both a more formal and a more general way. For this we first need the notion of a $+-S$-chain, which formalizes the notion of a potentially active chain of positive S-authorizations followed by a strong negative S-authorization, which can only be inactivated if attacked by another $+-S$-chain. The time stamp of a $+-S$-chain indicates the least time stamp that a positive authorization must have in order not to be affected by the $+-S$-chain.

Definition 3.5. *A* $+-S$-chain with time stamp t *is a chain of authorizations* $(p_0, p_1, +, S, t_1)$, $(p_1, p_2, +, S, t_2)$, $\ldots$, $(p_{n-1}, p_n, +, S, t_n)$, $(p_n, p_{n+1}, \tau, S, t_{n+1})$ *satisfying the following properties:*

- $p_0 = SOA$
- *Either* $\tau = -_{\mathrm{SR}}$ *and* $t = \infty$, *or* $\tau = -_{\mathrm{SN}}$ *and* $t = t_{n+1}$.
- *There are no* i, j *with* $0 \leq i < j \leq n$ *such that there is an authorization* $(p_i, p_j, -_{\mathrm{PR}}, S, t')$.
- *There are no* i, j *with* $0 \leq i < j \leq n$ *such that there is an authorization* $(p_i, p_j, -_{\mathrm{PN}}, S, t')$ *with* $t' < t_j$.

We say that a $+-S$-chain C_1 with time stamp t *attacks* a $+-S$-chain C_2 iff C_1 ends in a principal that has issued one of the authorizations in C_2 at some time $t' > t$.

Now the formalized and generalized constraint can be formulated as follows: We require that the $+-S$-chain in the authorization specification can be partially ordered in such a way that a $+-S$-chain C_1 attacks a $+-S$-chain C_2 only if $C_1 < C_2$ in the

partial ordering. Informally, this means that there should be no loops of $+-S$-chains under the attack relation. An authorization specification satisfying this constraint is called *free of strong S-revocation loops*.

3.5 Reasons for revocating

Hagström et al. have motivated the variety of revocation schemes by sketching various scenarios in which the principals performing the revocation have different reasons for revocating, so that different behaviour of the revocation is desirable. In order to study the desirable behaviour of the various revocation schemes, in this section we first present some scenarios, which we use to illustrate the different reasons the revocator has to perform the revocation, based on her level of trust or distrust towards the revokee. Having presented the scenarios, we informally sketch how we want to define the desirable behaviour of the various revocation schemes. These ideas will be formalized in subsequent sections.

3.5.1 Four scenarios

Scenario 1. *User A caught user C leaking information to a third-party. A revokes C's rights, ensuring that C cannot be given access by other users in the system.*

In this scenario user A had trusted user C in the past, thus issuing him an authorization, but now A distrusts principal C due the fact that he has leaked information to a third-party. So A will perform a *P-t-p Global Resilient* revocation, and—if she has strong revocation right—additionally a *Strong Global Resilient* revocation, both in order to remove the authorization she had granted and to forbid as many other principals as possible to grant new authorizations to C.

Scenario 2. *User C is leaving to join the rival company. When user A notices the situation, she preemptively blocks C's capabilities (but keeping the authorizations previously issued by C).*

In this scenario user A had trusted user C in the past, thus issuing him an authorization. Since C is leaving to the rival company, A now distrusts C to access files or to newly delegate access right to others, but since C never misused any rights, A still has trust in the delegation authorization previously issued by C. So A will perform a *P-t-p Local Resilient* revocation and—if possible—a *Strong Local Resilient* revocation, in order to remove the authorizations that had been granted to C and to forbid as many other principals as possible to grant new authorizations to C, at the same time preserving the effect of authorizations that C had previously delegated.

Scenario 3. *User A hears the rumour that user B has received a bribe, but A does not know whether the rumour is true. Upon informing other users, A revokes B's rights, allowing other users to re-issue them.*

In this scenario, A no longer trusts B since she has heard the rumour about B, so she will revoke B's rights. But since A does not know whether the rumour is true, A allows other users to give the rights back to B (A will tell others about the rumour and trusts them to only give the rights back to B if they know the rumour to be false). So A will perform a *P-t-p Global Non-resilient* revocation and – if possible – a *Strong Global Non-resilient* revocation, in order to inactivate the previously issued authorizations granted to B, at the same time allowing other users to newly grant authorizations to B.

Scenario 4. *User A is revising the authorizations she had granted in the past. During the process A finds an authorization to user C, whom A does not remember.*

In this scenario user A had trusted user C in the past, thus issuing him an authorization, but now A neither trusts nor distrusts C, as she has no recollection of who C is or why the authorization had been granted. So A will perform a *Weak Global Delete* in order to remove the authorization she had granted to C without affecting authorizations granted to C by other users.

3.5.2 Desirable behaviour of revocation schemes

When explaining the above scenarios, we referred to the level and manner of trust or distrust between the revoker and the revokee in order to motivate the choice of different revocation schemes in different scenarios. The main novel idea of this chapter is to formalize this reasoning about trust, delegation and revocation in such a way that we can formulate desirable properties that graph-theoretic definitions of revocation schemes should satisfy. Before we present this formalization, we will—for the rest of this section—first sketch the ideas behind this formalization and these desirable properties.

In Section 3.6, we will define *Trust Delegation Logic* (*TDL*), a logic that allows us to reason about the different levels and manners of trust or distrust that we find in the above four scenarios. One central idea in this logic is that A grants B the right to further delegate some right only if A trusts B to make correct judgments about who should be given that right. By expressing her trust in B to make correct judgments about something, A commits herselves to the truth of judgments that she has not made herself, namely the judgments that B has committed himself to. When A makes a judgment herself, we say that A has *explicit belief* in the judgment, whereas a judgment that A is committed to in the light of a principal trusted by A believing the statement is an *implicit belief* of A. Trust of principal A in principal B is modelled as A's belief in B's trustworthy. Depending on whether A's belief is explicit or implicit, we can also call this trust explicit or implicit. For example, if A expresses trust in B concerning the action of expressing trust in other principals, and B expresses trust in C, then A explicitly trusts B and implicitly trusts C.

A further central idea is that a principal A should have access right of access type α iff the SOA of that object trusts A, either explicitly or implicitly, concerning access α. Delegation chains correspond to chains of principals along whom an implicit trust in some principal can project upwards towards the SOA. A revocation takes place

when at some point along such a chain of principals, a principal stops trusting in the next principal on the chain, thus disabling this upward projection of implicit trust.

TDL allows us to model different ways in which a principal can stop trusting or start distrusting another principal. Some of these ways have been illustrated in the above four scenarios. The various revocation schemes correspond to these various ways of stopping to trust.

Given these explanations, we can now sketch how TDL allows us to formulate a desirable property for graph-theoretic definitions of revocation schemes: The graph-theoretic definitions of the revocation schemes should be such that for any given delegation and revocation interaction between the principals, an active authorization to a principal A should exist in a graph if and only if—translating the delegation and revocation behaviour to TDL—the SOA believes A to be trustworthy for the access in question.

3.6 A logic for reasoning about delegation and revocation

In this section we present a logic for formalizing the reasons for revoking delegations. This logic, which we call *Trust Delegation Logic* (*TDL*), is a first-order multi-modal logic with both classical negation and negation-as-failure. TDL formalizes both the relation of trust between principals and the action of announcing one's trust in another principal by delegating some right to him/her/it. In developing TDL, we have taken over some ideas from [Dem04] and [ABB+11].

We first define the syntax of TDL. Next we motivate its constructs and some of its axioms by sketching how we apply TDL for modelling delegation and revocation. After that we formally define the proof theory of TDL, briefly motivating the remaining axioms. TDL is only defined proof-theoretically, i.e. it does not have a formal semantics, since this is not needed for the application that we have in mind.

3.6.1 TDL syntax

There are five types of objects, *principal*, *access*, *time point*, *set of principal-time-pairs* and *announcement modality*. SOA is a constant symbol of type *principal*, $\emptyset$ is a constant symbol of type *set of principal-time-pairs*, B, $\not{B}$, K and $\not{K}$ are constants of type *announcement modality*, and ∞ and all integers are constants of type *time point*. *scons* is a ternary function symbol taking a term of type *principal*, a term of type *time* and a term of type *set of principal-time-pairs*, and returning a term of type *set of principal-time-pairs*. We use t, t', t_2, t_3 as *time point* variables, i, j, k as *principal* variables, $\Sigma, \Sigma', \Sigma_2$ as variables of type *set of principal-time-pairs*, α as an *action* variable, m as an *announcement modality* variable, and x as a variable of arbitrary type. Formulae of TDL are defined by the following EBNF rule:

$$\varphi ::= \neg\varphi \mid {\sim}\varphi \mid (\varphi \wedge \varphi) \mid \forall x\ \varphi \mid T_i\alpha \mid T_i^t\varphi \mid T_i\mathbb{D}\alpha \mid T_i\mathbb{S}\alpha \mid$$
$$B_{i,\Sigma}^t\varphi \mid K_{i,\Sigma}^t\varphi \mid I_i^t m\varphi \mid R_{i,\Sigma}^t m\varphi \mid r_i^t\alpha \mid t > t' \mid (i,t) \in \Sigma$$

We write $\varphi \to \psi$ for $\neg(\varphi \wedge \neg\psi)$, $\varphi \leftrightarrow \psi$ for $(\varphi \to \psi) \wedge (\psi \to \varphi)$, $\exists t > t'\ \varphi$ for $\exists t\ (t > t' \wedge \varphi)$, and $\exists x\ \varphi$ for $\neg\forall x\ \neg\varphi$. We drop brackets according to usual conventions. If φ is of the form $\neg\psi$, $\bar{\varphi}$ denotes ψ. Else $\bar{\varphi}$ denotes $\neg\varphi$.

While $\neg\varphi$ is the classical negation of φ, ${\sim}\varphi$ is negation as failure, i.e. ${\sim}\varphi$ is provable when φ is not provable.

$\emptyset$ refers to the empty set, and given a set Σ of principal-time-pairs, $scons(i,t,\Sigma)$ refers to the set $\Sigma \cup \{(i,t)\}$ (*scons* stands for *set constructor* [JJ99]). The following two axioms model this behaviour of $\emptyset$ and *scons*:

$$(\emptyset)\ \forall i\ \forall t\ \neg(i,t) \in \emptyset$$
$$(scons)\ \forall i,j,t,t',\Sigma\ ((i,t) \in scons(j,t',\Sigma)\ \leftrightarrow$$
$$(i = j \wedge t = t') \vee (i,t) \in \Sigma)$$

For the sake of readability, we abuse notation by using common set-theoretic notation in TDL formulas, writing for example $\Sigma \cup \{(i,t)\}$ instead of $scons(i,t,\Sigma)$, and $\{(i,t),(j,t')\}$ instead of $scons(j,t',scons(i,t,\emptyset))$.

3.6.2 Motivating TDL

The formula $r_i^t\alpha$ intuitively means that at time t, i has access right of access type α (the object o which may be accessed with access type α is not made explicit in TDL).

As [Dem04], we make a distinction between *belief* and *strong belief*: A principal who *believes* φ at time t (denoted $B_{i,\Sigma}^t\varphi$) has some justification for φ but believes that the justification might be wrong. A principal i who *strongly believes* φ at time t (denoted $K_{i,\Sigma}^t\varphi$) on the other hand believes that his/her/its justification for φ is correct. The Σ in the subscript of the belief operators indicates whether the belief is explicit or implicit (see Section 3.5.2 for this distinction): If Σ is $\emptyset$, it is explicit belief. If Σ is a non-empty set, the belief is implicit, and Σ indicates the principals who mediate this implicit belief together with the time points of their beliefs that mediate this belief. For example, if i trusts j at time t, j trusts k at time t', and k believes φ at time t_2, then i implicitly believes φ at time t, and this implicit belief is mediated by j and k through their beliefs at time t' and t_2 respectively: $B_{i,\{(j,t'),(k,t_2)\}}^t\varphi$.

Similarly as in [ABB+11], the fact that a principal i trusts a principal j on access α is formalized in TDL by a formula of the form $K_{i,\Sigma}^t T_j\alpha$. Here $T_j\alpha$ can be read intuitively as "j is trustworthy on access α". This way of formalizing the trust relation between two agents has the advantage of formally clarifying the difference between not trusting someone and actively distrusting someone, the first being formalized by $\neg K_{i,\Sigma}^t T_j\alpha$ (i.e. i lacks a strong belief about the trustworthiness of j), and the second by $K_{i,\Sigma}^t\neg T_j\alpha$ (i.e. i believes that j not trustworthy). Furthermore, weak distrust ($B_{i,\Sigma}^t\neg T_j\alpha$) is a useful formalization for the reserved kind of distrust that we have in Scenario 3.

TDL allows us to model five different levels of trust between a principal i and a principal j: *Strong trust*, where i strongly believes that j is trustworthy ($K^t_{i,\Sigma}T_j\alpha$), *weak trust* ($B^t_{i,\Sigma}T_j\alpha \wedge \neg K^t_{i,\Sigma}T_j\alpha$), *lack of trust* ($\neg B^t_{i,\Sigma}T_j\alpha \wedge \neg B^t_{i,\Sigma}\neg T_j\alpha$), *weak distrust* ($B^t_{i,\Sigma}\neg T_j\alpha \wedge \neg K^t_{i,\Sigma}\neg T_j\alpha$), and *strong distrust* ($K^t_{i,\Sigma}a\neg T_j\alpha$). The distinction between weak trust and lack of trust will not be relevant for modelling the reasoning about delegation and revocation, but the distinction between the remaining four levels of trust will be relevant.

Additionally to trust in someone on an action, the logic can also express epistemic trust: $K^t_{i,\Sigma}T^{t'}_j\varphi$ intuitively means that i trusts j not to make mistakes in judgements about the truth value of φ, if the judgement is made before time point t'. The time point at which the judgement was made needs to be considered in order to correctly model local revocations, in which an agent still trusts the authorizations previously produced by the revokee, but does not trust new authorizations issued by the revokee (see Scenario 2).

The action of granting to j the right to perform action α is modelled in TDL by the action of publicly announcing one's trust in j on action α. Whenever one makes a public announement, the announcement gets marked as an announcement of belief (B), strong belief (K), lack of belief ($\not{B}$) or lack of strong belief ($\not{K}$). [Dem04] uses the letter I for the action of informing someone, which is similar to the action of public announcement; so we have decided to use the letter I to denote public announcements: For example, $I^t_iK\varphi$ intuitively means that i publicly announces its strong belief in φ at time t. $I^t_iKT_j\alpha$ means that i announces its strong trust in j on action α, and corresponds to i issuing a positive authorization for j with permission α at time t. Performing a Weak Global Delete revocation for permission α is achieved by the public announcement $I^t_i\not{B}T_j\alpha$, with which i retracts its trust in j by announcing that it no longer believes j to be trustworthy.

If i wanted to give j the right to give any principal k the right to perform access α, i could achieve this by publicly announcing its strong trust in j concerning judgements about the trustworthiness of other principals: $I^t_iK\ \forall k\ T^\infty_jT_k\alpha$. If i was trusted by

the SOA to make such an announcement, then j would now be trusted by the SOA to announce its trust in any principal k on access α, i.e. to grant the right to perform access α to any principal. However, j would not yet be permitted to delegate to someone else the right to grant this right. To give j this right, the i's announcement would have to be $I_i^t K\ \forall k_1 T_j^\infty\ \forall k_2\ T_{k_1}^\infty T_{k_2}\alpha$. After this announcement, j can make an announcement of the form $I_j^{t'} K\ \forall k_2\ T_{k_1}^\infty T_{k_2}\alpha$, i.e. j can grant to some principal k_1 the right to grant to some further principal k_2 the right to perform access α.

This method can be used to model delegation with an arbitrary bound on the length for the delegation chain. But both Hagström et al.'s framework and our refinement of it do not put any bound on the length of delegation chains. In order to use this method for modelling delegation with no bound on the length of the delegation chain, we would have to allow principals to make infinitely many public announcements at once. In order to avoid this complication, we have introduced a third kind of trust, denoted $T_i\mathbb{D}\alpha$. Its intuitive meaning is that i is trusted to delegate the right to perform access α. Formally, its intended semantics is that $T_j\mathbb{D}\alpha$ should imply every formula in the infinite set $\{T_j\alpha,\ \forall k\ T_j^\infty T_k\alpha,\ \forall k_1 T_j^\infty\ \forall k_2\ T_{k_1}^\infty T_{k_2}\alpha,\ \dots\}$. This is achieved by the following axiom governing the behaviour of $T_i\mathbb{D}\alpha$:

$$(T\mathbb{D})\ T_i\mathbb{D}\alpha \rightarrow T_i\alpha \wedge T_i^\infty T_j\mathbb{D}\alpha$$

Using this new kind of trust, we can use the public announcement $I_i K T_j\mathbb{D}\alpha$ to model i's issuing a positive authorization for j with permission D at time t. $I_i^t \not\!\!B T_j\mathbb{D}\alpha$ models i's performing a Weak Global Delete revocation on j for permission $\mathbb{D}$.

Performing a P-t-p Global Resilient revocation can be modelled by announcing strong distrust in another principal: $I_i^t K\neg T_j\alpha$ or $I_i^t K\neg T_j\mathbb{D}\alpha$. If i explicitly announces its distrust j in this way, i thereby prevents an implicit belief in the trustworthiness of j to be passed through i to the SOA. Hence j will need to be connected to the SOA via some trust chain that is independent of i in order to get access or delegation right.

If i's strong distrust in j corresponds to a p-t-p revocation, what will correspond to a strong revocation? The answer is that for a strong revocation i needs to make an announcement that will ensure that the SOA does not trust j. For if the SOA is ensured not to trust j, then j's rights are blocked, just as after a strong revocation. Of course, in blocking j's rights in this way, i will have to make use of the fact that the SOA has—either directly or indirectly—granted i the right to perform a strong revocation. As a first attempt at modelling strong revocation and the right to perform strong revocations in TDL, it therefore makes sense to consider the following approach (for simplicity, we restrict ourselves to strong revocations for permission α):

Approach 1. *$I_i^t K \neg T_j \alpha$ models not only p-t-p revocation, but also strong revocation. The stronger effect of strong revocation is achieved by having the SOA believe in i's judgements of other principals non-trustworthiness when i has the strong revocation right. So i's issuing a positive authorization to j for permission S should be modelled by i announcing j to be trusted on distrusting other principals: $I_i^t K \forall k \; T_j^\infty \neg T_k \alpha$.*

The problem with this approach is that it would lead to blocked access in some situations where access should be granted. Suppose for example that the SOA grants A strong revocation right, A grants this right further to B, and B uses this right to issue a strong negative authorization to C. Furthermore, the SOA grants simple access right to A, who grants this further directly to C. So far, C does not have access, since its access is blocked by the strong negative authorization issued by B. But suppose next that the SOA globally revokes A's strong revocation right. Then B also loses its strong revocation right, so that the negative authorization issued by B becomes inactive. Hence C should now have access. But with the above approach, the fact that A granted B the strong revocation right means that A trusts B on distrusting other principals. Since B still distrusts C, this would mean that A implicitly distrusts C, so that C cannot have access based on a trust chain going through A.

To solve this problem, we model i's performing a Strong Global Resilient revocation on j for permission α by i announcing that the SOA should strongly

distrust j: $I_i^t K \forall t\ K^t_{\mathrm{SOA},\emptyset} \neg T_j \alpha$. Note that nested belief modalities are interpreted in a deontic way: $K^t_{i,\emptyset} K^t_{j,\emptyset} \varphi$ means that i strongly believes that j *should* strongly believe that φ. Granting i strong delegation right should make i being trusted on judgements about the SOA's strong distrust in other principals. In order to also be able to model performing a Strong Global Resilient revocation for permission S, we need to introduce a fourth kind of trust denoted $T_i \mathbb{S} \alpha$, for reason similar as the reasons for introducing $T_i \mathbb{D} \alpha$. The intuitive meaning of $T_i \mathbb{S} \alpha$ is that i is trusted in judgements about the SOA's strong distrust in other principals; here we need to allow for public announcements of distrust of various kinds: $\neg T_j \alpha$, $\neg T_j \mathbb{D} \alpha$ and $\neg T_j \mathbb{S} \alpha$. Furthermore, $T_i \mathbb{S} \alpha$ should imply that i is trusted to delegate the right to perform strong revocations, i.e. to consider another principal k trustworthy for performing strong revocations. The following axiom captures all this:

$$
\begin{aligned}
(T\mathbb{S})\ T_i \mathbb{S} \alpha \rightarrow\ & T_i^{\infty} T_j \mathbb{S} \alpha \wedge T_i^{\infty} \forall t\ K^t_{\mathrm{SOA},\emptyset} \neg T_j \alpha \wedge \\
& T_i^{\infty} \forall t\ K^t_{\mathrm{SOA},\emptyset} \forall k\ \forall t_2\ \neg T_j^{t_2} T_k \mathbb{D} \alpha \wedge T_i^{\infty} \forall t\ K^t_{\mathrm{SOA},\emptyset} \neg T_j \mathbb{S} \alpha
\end{aligned}
$$

A principal i may epistemically trust both a principal who believes φ and a principal who believes $\neg\varphi$. In such a situation we do not want i to implicitly hold the inconsistent beliefs that φ and that $\neg\varphi$, because we want implicit belief to stay consistent. Instead, we want i to implicitly believe neither φ nor $\neg\varphi$. So the principle that i's epistemic trust in j concerning φ and j's belief in φ together imply i's implicit belief in φ cannot hold without exception. Instead, we say that if i epistemically trusts j concerning φ and j believes φ, then i has a *reason* to believe φ. To deduce that i believes φ from the fact that i has a reason to believe φ we additionally require there to be no reason for i to believe $\neg\varphi$. In TDL, $R^t_{i,\Sigma} B \varphi$ (respectively $R^t_{i,\Sigma} K \varphi$) denotes the fact that at time t, i has a reason to believe (respectively to strongly believe) φ implicitly, mediated by Σ. In order for the absence of a reason for i to believe $\neg\varphi$ to be provable, it needs to be formulated using negation-as-failure rather than classical negation: $\sim \exists \Sigma\ R^t_{i,\Sigma} B \neg \varphi$.

3.6.3 TDL proof theory

In order to correctly capture the intended functioning of negation-as-failure in TDL's proof theory, we need TDL's deducibility relation $\Gamma \vdash \varphi$ to be defined in such a way that $\Gamma \nvdash \varphi$ in general implies $\Gamma \vdash \sim\varphi$. This can be achieved by defining this deducibility relation inductively[a] as follows:

Definition 3.6. *We define $\Gamma \vdash \varphi$ to be the case if one of the following conditions holds:*

- $\varphi \in \Gamma$
- *φ is an axiom of TDL*
- *For some formula ψ, $\Gamma \vdash \psi$ and $\Gamma \vdash \psi \rightarrow \varphi$ (modus ponens)*
- *φ is of the form $K^t_{i,\Sigma}\psi$ and $\vdash \psi$ (necessitation for strong belief)*
- *φ is of the form $R^t_{i,\Sigma}K\psi$ and $\vdash \psi$ (necessitation for reasons for strong belief)*
- *φ is of the form $\sim\psi$, where ψ is not of the form $\sim\chi$, and $\Gamma \nvdash \psi$ (negation-as-failure)*

The axioms of TDL include the axioms of the standard Hilbert system for first-order logic (as described for example in subchapter 3.6 of [Rau06]) as well as all axioms mentioned in section 3.6.2 and in the rest of this section.

The axioms governing the behaviour of the two belief modalities and their interaction are taken over from Demelombe [Dem04]. Both belief modalities obey the system (KD):

$$(\mathrm{K}_B)\ B^t_{i,\Sigma}\varphi \wedge B^t_{i,\Sigma}(\varphi \rightarrow \psi) \rightarrow B^t_{i,\Sigma}\psi$$

$$(\mathrm{D}_B)\ \neg(B^t_{i,\Sigma}\varphi \wedge B^t_{i,\Sigma}\neg\varphi)$$

$$(\mathrm{K}_K)\ K^t_{i,\Sigma}\varphi \wedge K^t_{i,\Sigma}(\varphi \rightarrow \psi) \rightarrow K^t_{i,\Sigma}\psi$$

$$(\mathrm{D}_K)\ \neg(K^t_{i,\Sigma}\varphi \wedge K^t_{i,\Sigma}\neg\varphi)$$

[a] Since Definition 3.6 is also an inductive definition, it can–similarly to Definitions 3.2 and 3.3 discussed in Section 3.4–in some contexts lead to the relation $\vdash$ being undefined. However, as follows from the proof of Theorem 3.1, $\Gamma \vdash \varphi$ is defined whenever Γ is a set of announcement formulas corresponding to a authorization specification free of strong S-revocation loops.

Furthermore, strong belief satisfies the axiom schema (KT), which intuitively says that a principal strongly believes that what it strongly believes is true:

$$(\mathrm{KT})\ K^t_{i,\Sigma}(K^t_{i,\emptyset}\varphi \to \varphi)$$

Strong belief implies weak belief:

$$(KB)\ K^t_{i,\Sigma}\varphi \to B^t_{i,\Sigma}\varphi$$

We need axioms similar to these axioms about the two belief modalities for the reason-for-belief modality:

$$\begin{aligned}
(\mathrm{K}_{RB})\ & R^t_{i,\Sigma}B\varphi \wedge R^t_{i,\Sigma}B(\varphi \to \psi) \to R^t_{i,\Sigma}B\psi \\
(\mathrm{K}_{RK})\ & R^t_{i,\Sigma}K\varphi \wedge R^t_{i,\Sigma}K(\varphi \to \psi) \to R^t_{i,\Sigma}K\psi \\
(\mathrm{KT}_R)\ & R^t_{i,\Sigma}KK^t_{i,\emptyset}K\varphi \to R^t_{i,\Sigma}K\varphi \\
(RKRB)\ & R^t_{i,\Sigma}K\varphi \to R^t_{i,\Sigma}B\varphi
\end{aligned}$$

Recall that $B^t_{i,\Sigma}B^{t'}_{j,\Sigma'}\varphi$ is interpreted to mean that i believes that j *should* believe that φ. It is reasonable to assume that i believes that someone else should believe φ iff i herself believes φ. This is captured by the following four axiom schemas:

$$\begin{aligned}
(BB_1)\ & B^t_{i,\Sigma}\varphi \to B^t_{i,\Sigma}\forall t' B^{t'}_{j,\emptyset}\varphi \\
(BB_2)\ & B^t_{i,\Sigma}B^{t'}_{j,\emptyset}\varphi \to B^t_{i,\Sigma}\varphi \\
(KK_1)\ & K^t_{i,\Sigma}\varphi \to K^t_{i,\Sigma}\forall t' K^{t'}_{j,\emptyset}\varphi \\
(KK_2)\ & K^t_{i,\Sigma}K^{t'}_{j,\emptyset}\varphi \to K^t_{i,\Sigma}\varphi
\end{aligned}$$

The action of asserting a strong belief is always also considered an action of asserting the corresponding weak belief, and the action of denying a weak belief

is always also considered an action of denying the corresponding strong belief:

$$(IKB)\ I_i^t K\varphi \rightarrow I_i^t B\varphi$$
$$(IBK)\ I_i^t \not{B}\varphi \rightarrow I_i^t \not{K}\varphi$$

Since the SOA has the ultimate authority over the object in question, every principal has the right to perform an action iff it is trusted by the SOA on that action:

$$(\text{SOA})\ r_i^t\alpha \leftrightarrow \exists\Sigma\ K^t_{\text{SOA},\Sigma} T_i\alpha$$

If at time t, i has epistemic trust in j concerning the judgements about φ made before time point t', this means that if j believes φ at time $min(t,t')$, i generally has a reason to believe φ. However, this reason to believe φ cannot be inferred if j believes φ only implicitly, mediated by the belief of some principal k at time t, and i has a reason to distrust k concerning judgements about φ held at time t. This is formalized in the *axiom schemas of epistemic trust*:

$$\begin{aligned}(\text{ET}_B)\ & B^t_{i,\emptyset} T^{t'}_j \varphi \wedge ((t' > t \wedge t_2 = t) \vee (\neg t' > t \wedge t_2 = t')) \wedge \\ & B^{t_2}_{j,\Sigma}\varphi \wedge {\sim}\exists k, t_3, \Sigma'\ ((k,t_3) \in \Sigma \wedge R^t_{i,\Sigma'} B \neg T^{t_3}_k \varphi) \\ & \rightarrow\ R^t_{i,\Sigma\cup\{(j,t_2)\}} B\varphi \\ (\text{ET}_K)\ & K^t_{i,\emptyset} T^{t'}_j \varphi \wedge ((t' > t \wedge t_2 = t) \vee (\neg t' > t \wedge t_2 = t')) \wedge \\ & K^{t_2}_{j,\Sigma}\varphi \wedge {\sim}\exists k, t_3, \Sigma'\ ((k,t_3) \in \Sigma \wedge R^t_{i,\Sigma'} B \neg T^{t_3}_k \varphi) \\ & \rightarrow\ R^t_{i,\Sigma\cup\{(j,t_2)\}} K\varphi\end{aligned}$$

The following axioms govern the relationship between belief and reason to belief that we already explained at the end of Section 3.6.2:

$$(RB)\ R^t_{i,\Sigma} B\varphi \wedge {\sim}\exists \Sigma'\ R^t_{i,\Sigma'} B\neg\varphi \to B^t_{i,\Sigma}\varphi$$
$$(RK)\ R^t_{i,\Sigma} K\varphi \wedge {\sim}\exists \Sigma'\ R^t_{i,\Sigma'} B\neg\varphi \to K^t_{i,\Sigma}\varphi$$
$$(BR)\ B^t_{i,\Sigma}\varphi \to R^t_{i,\Sigma} B\varphi$$
$$(KR)\ K^t_{i,\Sigma}\varphi \to R^t_{i,\Sigma} K\varphi$$

We assume that principals are sincere, in the sense that in general a principal believes what he/she/it has previously announced. However, this principle needs some restrictions: Firstly, a principal can distance itself from a previous annoncement by making an announcement with the same content as before but with opposite announcement modality (e.g., to distance itself from its previous announcement of belief in φ, a principal can announce its non-belief in φ). Secondly, an announcement of weak belief in φ can be made obsolete by an announcement of strong belief in $\bar{\varphi}$ by a trustworthy principal. Thirdly, an announcement of strong belief only implies that the principal has reasons for strong belief; strong belief can be implied using axiom (RK) in the absence of reasons for the negation. This is formalized in the *sincerity axiom schemas*:

$$(\mathrm{Sin}_B)\ I^{t'}_i B\varphi \wedge t > t' \wedge {\sim}\exists t_2 > t'\ (t > t_2 \wedge I^{t_2}_i \not{B}\varphi) \wedge$$
$${\sim}\exists j, \Sigma\ \exists t_3 > t'\ (t > t_3 \wedge B^{t_3}_{i,\Sigma} T^t_j \bar{\varphi} \wedge I^{t_3}_j K\bar{\varphi}) \to B^t_{i,\emptyset}\varphi$$
$$(\mathrm{Sin}_K)\ I^{t'}_i K\varphi \wedge t > t' \wedge\ {\sim}\exists t_2 > t'\ (t > t_2 \wedge I^{t_2}_i \not{K}\varphi) \to R^t_{i,\emptyset} K\varphi$$
$$(\mathrm{Sin}_{\not{B}})\ I^{t'}_i \not{B}\varphi \wedge t > t' \wedge\ {\sim}\exists t_2 > t'\ (t > t_2 \wedge I^{t_2}_i B\varphi) \to \neg B^t_{i,\emptyset}\varphi$$
$$(\mathrm{Sin}_{\not{K}})\ I^{t'}_i \not{K}\varphi \wedge t > t' \wedge\ {\sim}\exists t_2 > t'\ (t > t_2 \wedge I^{t_2}_i K\varphi) \to \neg K^t_{i,\emptyset}\varphi$$

We assume that all principals trust themselves, as stated by the *axiom of self-trust*:

$$(\mathrm{ST})\ K^t_{i,\emptyset} T_i \alpha$$

Since $T_k^t\varphi$ means that k's judgements made before time point t about φ are trusted, $T_k^t\varphi$ should clearly imply $T_k^{t'}\varphi$ if t' is an earlier time point than t:

$$(T^t)\ T_k^t\varphi \wedge t > t' \rightarrow T_k^{t'}\varphi$$

In order for the binary relation $>$ to function properly in the logic, we need the following axiom scheme. For any two time point constants $c_1, c_2 \in \mathbb{Z} \cup \{\infty\}$, if $c_1 < c_2$ in the natural ordering of $\mathbb{Z} \cup \{\infty\}$ (in which ∞ is larger than any integer), the following formula is a TDL axiom:

$$(>)\ c_1 < c_2 \wedge \neg c_2 < c_1 \wedge \neg c_1 < c_1$$

3.7 Scenarios in TDL

In this section we show how TDL can be used to model the reasoning about trust and distrust involved in justifying the choices of revocation schemes in the scenarios from section 3.5.1.

In order to formalize scenario 3, we need to add some details to the description of the scenario: Suppose that A is the SOA and that at time point 1, A grants C delegation right concerning the access α, i.e. $I_{\mathrm{A}}^1 K T_{\mathrm{C}} \mathbb{D}\alpha$. At time 2, C grants B this delegation right: $I_{\mathrm{C}}^2 K T_{\mathrm{B}} \mathbb{D}\alpha$. Later, let's say at time point 9, A finds out that C is leaving to join the rival company, and hence now distrusts C concerning access α or to grant delegation right concerning access a to anyone else: $I_{\mathrm{A}}^9 K \neg T_{\mathrm{C}}\alpha$ and $I_{\mathrm{A}}^9 K \neg T_{\mathrm{C}} \mathbb{D}\alpha$. A also explicitly denies her previous trust statement to make clear it is no longer in place: $I_{\mathrm{A}}^9 \not{K} T_{\mathrm{C}} \mathbb{D}\alpha$. But since C never misused any rights, A still trusts the delegation authorizations issued by C before time point 9: $I_{\mathrm{A}}^9 K\ \forall k\ T_{\mathrm{C}}^9 T_k \mathbb{D}\alpha$. We expect that C loses his access and delegation rights at time point 9, but that B retains these rights.

We now explain how this expected result is actually attained in TDL: By axiom (Sin_K), $I_{\mathrm{A}}^1 K T_{\mathrm{C}} \mathbb{D}\alpha$ and the fact that A does not deny this announcement before time

point 9 imply that for $1 \leq t \leq 8$, $K^t_{\mathrm{A},\emptyset} T_{\mathrm{C}} \mathbb{D}\alpha$, which by $(T\mathbb{D})$ and (K_K) further implies $K^t_{\mathrm{A},\emptyset} T_{\mathrm{C}} \alpha$. Since $A = \mathrm{SOA}$, axiom (SOA) implies $r^t_C \alpha$ and $r^t_C \mathbb{D}\alpha$, i.e. C has access and delegation rights from time point 2 until time point 8. But since $I^9_{\mathrm{A}} \not{K} T_{\mathrm{C}} \mathbb{D}\alpha$, we cannot deduce $r^9_C \alpha$ and $r^9_C \mathbb{D}\alpha$ in this way: At time point 9, C no longer has access and delegation right, as expected. However, for $2 \leq t \leq 9$, we can deduce $r^t_B \alpha$ and $r^t_B \mathbb{D}\alpha$, i.e. that B has access and delegation rights: First, note that for $1 \leq t \leq 9$ we have $K^t_{\mathrm{A},\emptyset}\ \forall k\ T^9_{\mathrm{C}} T_k \mathbb{D}\alpha$ (which by (K_K) implies $K^t_{\mathrm{A},\emptyset} T^9_{\mathrm{C}} T_{\mathrm{B}} \mathbb{D}\alpha$). In case $1 \leq t \leq 8$, this follows from $K^t_{\mathrm{A},\emptyset} T_{\mathrm{C}} \mathbb{D}\alpha$, $(T\mathbb{D})$ and (K_K); in case $t = 9$, it follows from $I^9_{\mathrm{A}} K\ \forall k\ T^9_{\mathrm{C}} T_k \mathbb{D}\alpha$ and (Sin_K). $I^2_{\mathrm{C}} K T_{\mathrm{B}} \mathbb{D}\alpha$ and (Sin_K) imply that for $2 \leq t \leq 9$, $K^t_{\mathrm{C}} T_{\mathrm{B}} \mathbb{D}\alpha$, so using (ET_K), we can derive $K^t_{\mathrm{A},\emptyset} T_{\mathrm{B}} \mathbb{D}\alpha$, which similarly as in the above proofs of $r^t_C \alpha$ and $r^t_C \mathbb{D}\alpha$ implies $r^t_B a$ and $r^t_B \mathbb{D}\alpha$.

Here is how the other scenarios discussed in Section 3.5 can be formalized in TDL (we assume that the revocation always takes place at time point 9):

Scenario 1. User A distrusts user C concerning access and delegation: $I^9_{\mathrm{A}} K \neg T_{\mathrm{C}} \mathbb{D}\alpha$. This implies not only $K^9_{\mathrm{A},\emptyset} \neg T_{\mathrm{C}} \mathbb{D}\alpha$, but by axiom (KK_1) also $K^9_{\mathrm{A},\emptyset} \forall t\ K^t_{\mathrm{SOA},\emptyset} \neg T_{\mathrm{C}} \mathbb{D}\alpha$. According to the explanations about strong revocations in section 3.6.2, if the SOA trusts A on strong revocations ($\exists \Sigma\ K^9_{\mathrm{SOA},\Sigma} T_{\mathrm{A}} \mathbb{S}\alpha$), the latter formula has the same effect as a Strong Global Resilient revocation.

Scenario 3. The reserved kind of distrust resulting from hearing a rumour for which one does not know whether it is true is modelled in TDL as weak distrust, i.e. weak belief in the non-trustworthiness of the principal in question: $I^9_{\mathrm{A}} B \neg T_{\mathrm{B}} \mathbb{D}\alpha$. Since axiom (Sin_B) blocks the inference of $B^t_{i,\emptyset} B \neg T_{\mathrm{B}} \mathbb{D}\alpha$ from $I^9_{\mathrm{A}} B \neg T_{\mathrm{B}} \mathbb{D}\alpha$ if some trusted principal announces trust in j (i.e. delegates to j), $I^9_{\mathrm{A}} B \neg T_{\mathrm{B}} \mathbb{D}\alpha$ loses its effect as soon as such an anouncement takes place. Hence we have the effect of a non-resilient revocation, as desired.

Scenario 4. User A neither trusts nor distrusts user C: $I^9_{\mathrm{A}} \not{B} T_{\mathrm{C}} \mathbb{D}\alpha$ and $I_{\mathrm{A}} \not{B}^9 \neg T_{\mathrm{C}} \mathbb{D}\alpha$. These announcements remove the effect of any previous announcement made by A about the trustworthiness of C, i.e. the situation is now practically the

same as if A had never trusted C. This corresponds to a Weak Global Delete, in which a positive authorization is removed, leaving no trace of it ever having been there.

3.8 Desirable behaviour of revocation schemes

In this section we show how TDL can be used to formally formulate a desirable property for a graph-theoretically defined revocation framework. This allows us to study revocation frameworks using the axiomatic method originating in social choice theory.

There are different revocation schemes because there are different reasons for revocating. We start this section by exhibiting a correspondence between revocation schemes and reasons for revocating formalizable in TDL. The main idea behind the desirable property of revocation frameworks that we define is that if performing revocation schemes and granting rights was replaced by publicly announcing one's formal reasons for revocating or granting, then these public announcements should logically imply (in TDL) a principal's access right iff that principal is actually granted access based on the delegation graph.

3.8.1 Matching reasons for revocating to revocation schemes

As explained in Section 3.6, there are five levels of trust that an agent can have in another agent, of which four need to be distinguished in modelling delegation and revocation. But even when i explicitly strongly distrusts j concerning delegation right ($K^t_{i,\emptyset}\neg T_j\mathbb{D}\alpha$), i may still trust j's previous judgements concerning $T_k\mathbb{D}\alpha$ for other principals k ($K^t_{i,\emptyset}\forall k T^t_j T_k\mathbb{D}\alpha$). So the level of trust in another agent concerning delegation right can be different from the level of trust concerning previously granted authorizations. However, these two levels of trust are not completely independent of each other: For example, $K^t_{i,\emptyset}T_j\mathbb{D}\alpha$ implies $K^t_{i,\emptyset}\forall k T^t_j T_k\mathbb{D}\alpha$. More generally, the

second level of trust must be at least as high as the first level of trust. This means that only 10 of the 16 ensuing combinations of trust levels are actually possible.

Table 3.1 shows which granting-revocation behaviour corresponds to each of these ten possible combinations of trust levels. Some cells contain multiple revocation schemes. This means that the granting-revocation behaviour corresponding to the combination of trust levels represented by that cell consists of performing multiple revocation schemes at the same time. For an agent without strong revocation rights, the granting-revocation behaviour corresponding to some combination of trust levels is determined by dropping the strong revocations from the revocations in the cell that represent that combination of trust levels.

The formulas in the table have $\bar{\pi}$ in place of α, $\mathbb{D}\alpha$ or $\mathbb{S}\alpha$. $\bar{\pi}$ is defined as follows:

Definition 3.7. *For $\pi \in \{A, D, S\}$, we define $\bar{\pi}$ by setting $\bar{\pi} := \alpha$ if $\pi = A$, $\bar{\pi} := \mathbb{D}\alpha$ if $\pi = D$, and $\bar{\pi} := \mathbb{S}\alpha$ if $\pi = S$.*

The revocation schemes in the table should always be for the same π that is used in the $\bar{\pi}$ in the formulas.

We consider the pair of levels of trust that i has in j to be the reason i has for granting a right to j or revoking a right from j. Hence the graph-theoretic definitions of the revocation schemes should be such that access is granted whenever this is justifiable on the basis of these trust-based reasons for granting and revocating. We use deducibility in TDL as our formal criterion for justifiablity.

These explanations already determine the desirable property for a set of graph-theoretic definitions of revocation schemes. We now proceed to formalizing this desirable property.

3.8.2 Formal desirable property

We first define a set $\mathbf{C}$ corresponding to the ten meaningful cells of Table 3.1:

Definition 3.8. $\mathbf{C} := \{(m, n) \in \{1, 2, 3, 4\}^2 \mid m \geq n\}$.

	$K^t_{i,\emptyset}\forall k T^t_j T_k \bar{\pi}$	$\neg B^t_{i,\emptyset}\forall k T^t_j T_k \bar{\pi} \wedge$ $\neg B^t_{i,\emptyset}\forall t' \forall k \neg T^{t'}_j T_k \bar{\pi}$	$B^t_{i,\emptyset}\forall t' \forall k \neg T^{t'}_j T_k \bar{\pi} \wedge$ $\neg K^t_{i,\emptyset}\forall t' \forall k \neg T^{t'}_j T_k \bar{\pi}$	$K^t_{i,\emptyset}\forall t' \forall k \neg T^{t'}_j T_k \bar{\pi}$
$K^t_{i,\emptyset} T_j \bar{\pi}$	grant permission π	X	X	X
$\neg B^t_{i,\emptyset} T_j \bar{\pi} \wedge \neg B^t_{i,\emptyset} \neg T_j \bar{\pi}$	WLD	WGD	X	X
$B^t_{i,\emptyset} \neg T_j \bar{\pi} \wedge \neg K^t_{i,\emptyset} \neg T_j \bar{\pi}$	PLN $\circ$ SLN	WGD $\circ$ PLN $\circ$ SLN	PGN $\circ$ SGN	X
$K^t_{i,\emptyset} \neg T_j \bar{\pi}$	PLR $\circ$ SLR	WGD $\circ$ PLR $\circ$ SLR	PGN $\circ$ PLR $\circ$ SGN $\circ$ SLR	PGR $\circ$ SGR

Table 3.1: The correspondence between the revocation framework and reasons for revocating formalized in TDL

Next we define a *granting-revocation action* corresponding to a cell in the table:

Definition 3.9. *For $i, j \in \mathbf{S}$, $(m, n) \in \mathbf{C}$ and $\pi \in \mathbf{P}$, define $\mathbf{GR}(i, j, (m, n), \pi)$ to be the granting-revocation behavior in the cell in row m and column n of Table 1 performed by i onto j for permission π.*

For example, $\mathbf{GR}(\mathrm{A}, \mathrm{B}, (2, 2), D)$ is a Weak Global Delete revocation from A to B for permission D.

Next we define a *public announcement* corresponding to a cell in the table:

Definition 3.10. *For $i, j \in \mathbf{S}$, $(m, n) \in \mathbf{C}$ and $\pi \in \mathbf{P}$, define $\mathbf{I}(i, j, (m, n), \pi)$ to be the set of public announcements by i in trust in j for permission π according to the level of trust of row m and column n of Table 1.*

For example, $\mathbf{I}(\mathrm{A}, \mathrm{B}, (4, 1), \alpha)$ is $\{I^t_{\mathrm{A}} K \forall k T^t_{\mathrm{B}} T_k \bar{\pi}, I^t_{\mathrm{A}} K \neg T_{\mathrm{B}} \bar{\pi}\}$.

We now need to define the notion of an authorization specification resulting from a sequence of granting-revocation-actions.

Definition 3.11. *Given a sequence σ of elements of $\mathbf{S} \times \mathbf{S} \times \mathbf{C} \times \mathbf{P}$, we define the authorization specification $\mathbf{A}(\sigma)$ inductively as follows:*

- $\mathbf{A}(\langle\rangle) = \emptyset$
- $\mathbf{A}(\langle (i_1, j_1, a_1, r_1), \ldots, (i_n, j_n, a_n, r_n) \rangle$ *is the authorization specification resulting from performing* $\mathbf{GR}(i_n, j_n, a_n, r_n)$ *on* $\mathbf{A}(\langle (i_1, j_1, a_1, r_1), \ldots, (i_{n-1}, j_{n-1}, a_{n-1}, r_{n-1}) \rangle)$.

Now we need to define which sequences of granting-revocation-actions are actually valid in our system:

Definition 3.12. *A sequence $\langle (i_1, j_1, a_1, r_1), \ldots, (i_n, j_n, a_n, r_n) \rangle$ of elements of $\mathbf{S} \times \mathbf{S} \times \mathbf{C} \times \mathbf{P}$ is called a* valid granting-revocation pattern *iff for every $k \leq n$, the authorization specification $\mathbf{A}(\langle (i_1, j_1, a_1, r_1), \ldots, (i_k, j_k, a_k, r_k) \rangle)$ is free of strong S-revocation loops and authorizes i_k to perform $\mathbf{GR}(i_k, j_k, a_k, r_k)$.*

Now we need to define the notion of a π-chain for $\pi \in \mathbf{P}$:

Definition 3.13. *A* π-chain *is a chain of authorizations* $(p_0, p_1, +, \pi', t_1)$, $(p_1, p_2, +, \pi', t_2)$, $\ldots$, $(p_{n-1}, p_n, +, \pi', t_n)$, $(p_n, p_{n+1}, +, \pi, t_{n+1})$ *satisfying the following properties:*

- $p_0 = SOA$
- $\pi' = D$ *if* $\pi = A$, *and* $\pi' = \pi$ *otherwise.*
- *There are no* i, j *with* $0 \leq i < j \leq n+1$ *such that there is an authorization* $(p_i, p_j, -_{\mathrm{PR}}, \pi'', t')$, *where* $\pi'' = \pi$ *if* $j = n+1$, *and* $\pi'' = \pi'$ *otherwise.*
- *There are no* i, j *with* $0 \leq i < j \leq n+1$ *such that there is an authorization* $(p_i, p_j, -_{\mathrm{PN}}, \pi'', t')$ *with* $t' < t_j$, *where* $\pi'' = \pi$ *if* $j = n+1$, *and* $\pi'' = \pi'$ *otherwise.*

The following theorem formally expresses that our refined revocation framework has the desirable property that we have previously already explained and motivated:

Theorem 3.1. *Let* $n \in \mathbb{N}$, *and let* σ *be a valid granting-revocation pattern of length* n. *Then for all* $i \in \mathbf{S}$, $\mathbf{I}(\sigma) \models r_i^n \alpha$ *iff* i *is the SOA or there is an active authorization of the form* $(p, i, +, \alpha, t)$ *in* $\mathbf{A}(\sigma)$.

Proof Sketch. {Proof Sketch by Marcos Cramer} We present a proof sketch by first exhibiting a procedure for determining which authorizations in $\mathbf{A}(\sigma)$ are active, and then exhibiting a correspondence between statements about the activeness of authorization in $\mathbf{A}(\sigma)$ and statements about the deducibility of certain TDL formulas from $\mathbf{I}(\sigma)$.

Note that a $+-S$-chain is an S-chain followed by negative S-authorization.

σ is a valid granting-revocation pattern, so $\mathbf{A}(\sigma)$ is free of strong S-revocation loops, i.e. there is a partial ordering $<$ on the set of $+-S$-chains over $\mathbf{A}(\sigma)$. Since the set of $+-S$-chains over $\mathbf{A}(\sigma)$ is finite, $<$ is a well-ordering, and we can perform induction along $<$. This allows us to determine which $+-S$-chains are active: A $+-S$-chain C_1 is active iff it is not attacked by an active $+-S$-chain $C_2 < C_1$.

Next we can establish which positive S-authorizations are active: A positive S-authorization is directly inactivated iff its end node is attacked by an active $+-S$-chain. A positive S-authorization is active iff it is an element of an S-chain whose authorizations are not directly inactivated.

Next we can establish that a strong negative authorizations is active iff it starts at some principal at which some active S-chain ends. This allows us to establish which other authorizations are directly inactivated: $(i, j, +, \pi, t)$ is directly inactivated iff there is an active authorization (k, j, τ, π, t') such that either $\tau = -_{\mathrm{SR}}$ or $\tau = -_{\mathrm{SN}}$ and $t < t'$. Finally, we can establish that a positive π-authorization is active iff it is an element of a π-chain whose authorizations are not directly inactivated.

In a similar way as one can show in a step-by-step way the correctness of this procedure for determining the activeness of authorizations in $\mathbf{A}(\sigma)$, one can prove the following three equivalences:

1. For $\tau = -_{\mathrm{SR}}$ or $\tau = -_{\mathrm{SN}}$, (i, j, τ, π, t) is active at time t' iff $\mathbf{I}(\sigma) \vdash \exists\Sigma\ K^{t'}_{\mathrm{SOA},\Sigma} T_i \mathbb{S}\alpha$.
2. $(i, j, +, \pi, t)$ is directly inactivated at time t' iff $\mathbf{I}(\sigma) \vdash \exists\Sigma\ R^{t'}_{\mathrm{SOA},\Sigma} B \neg T_j \bar{\pi}$.
3. $(i, j, +, \pi, t)$ is active at time t' iff $\mathbf{I}(\sigma) \vdash \exists\Sigma\ K^{t'}_{\mathrm{SOA},\Sigma} T_i \bar{\pi}$ and $\mathbf{I}(\sigma) \nvdash \exists\Sigma\ R^{t'}_{\mathrm{SOA},\Sigma} B \neg T_j \bar{\pi}$.

The theorem now follows from equivalence 3 together with TDL axioms (SOA), (RK) and (ST). □

Note that if we had refrained from implementing in our revised framework one of the five changes to Hagström et al.'s framework discussed in section 3.3.1, the resulting framework would not satisfy this desirably property.

3.9 Conclusion

After identifying some problems with Hagström et al.'s [HJPPW01] revocation framework, we presented a refined framework that avoids these problems. In order to ensure that our refined framework does not itself suffer from similar problems, we systematically studied the relation between the reasons for revocating and the graph-theoretic definitions of revocation schemes. In order to formalize reasons for revocating based on trust and distrust, we developed Trust Delegation Logic

(TDL). TDL allowed us to formulate a desirable property that a graph-theoretically defined revocation framework should satisfy. This desirable property is based on a correspondence between revocation schemes and reasons for revocating, and requires the revocation schemes to be defined in such a way that access is granted whenever this is justifiable on the basis of the reasons for granting and revocating. The main theorem of the chapter asserts that our refined framework does satisfy this desirable property.

Chapter 4

A Query-Driven Decision-Procedure for Distributed Autoepistemic Logic with Inductive Definitions

Abstract. In this chapter we first give a summary of autoepistemic logic (*AEL*) and distributed autoepistemic logic (*dAEL*). *dAEL* is a multiagent non-monotonic epistemic logic [VHCBD16]. We motivate its use for access control and define a query-driven decision procedure for the well-founded semantics of dAEL(ID) (distributed autoepistemic logic with inductive definitions). The decision procedure is designed in such a way that it allows to determine access rights while minimizing the information flow between principals in order to enhance security and reduce privacy concerns.

4.1 Motivation

Many logics have been proposed for dealing with *access control (AC)*. Some approaches focus on a more 'centralized' setting, e.g. assuming a central reference monitor. Other approaches focus on a distributed setting i.e. there is no central

monitoring. We will focus our study on the second case on which we do not assume any kind of centralization.

Most of these logics use a modality $k\ says$ indexed by a *principal* k. In this context a principal can be a process, organization, user, etc, i.e. any entity that can manage a resource or issue statements about the system's resources/artifacts. In the context of modal logics we can replace the concept of principal with *agent*, and may use these terms interchangeably.

Says-based access control logics are designed for systems in which different principals can issue statements that become part of the access control policy. The statement $k\ says\ \varphi$ is usually understood as "principal k supports statement φ", which can be interpreted to mean that principal k has issued statements that—together with some additional information present in the system—imply φ. Different access control logics vary in their account of which additional information may be assumed in deriving the statements that k supports.

Many state-of-the-art *says*-based access control logics, e.g. Deepak Garg's BL [GP12], do not provide the means for deriving statements of the form $\neg k\ says\ \varphi$ or $j\ says\ (\neg k\ says\ \varphi)$. However, being able to derive statements of the form $\neg k\ says\ \varphi$ and $j\ says\ (\neg k\ says\ \varphi)$ makes it possible to model access denials naturally in a *says*-based access control logic: Suppose A is a professor with control over a resource r, B is a PhD student of A who needs access to r, and C is a postdoc of A supervising B. A wants to grant B access to r, but wants to grant C the right to deny B's access to r, for example in case B misuses her rights. A natural way for A to do this using the *says*-modality is to issue the statement $(\neg C\ says\ \neg access(B, r)) \Rightarrow access(B, r)$. This should have the effect that B has access to r unless C denies her access. However, this effect can only be achieved if our logic allows A to derive $\neg C\ says\ \neg access(B, r)$ from the fact that C has not issued any statements implying $\neg access(B, r)$.

The derivation of $\neg C\ says\ \neg access(B, r)$ from the fact that C has not issued any statements implying $\neg access(B, r)$ is non-monotonic: If C issues a statement implying $\neg access(B, r)$, the formula $\neg C\ says\ \neg access(B, r)$ can no longer be derived. In other

words, adding a formula to the access control policy causes that something previously implied by the policy is no longer implied. Existing *says*-based access control logics are monotonic, so they cannot support the reasoning described above for modelling denial with the *says*-modality.

In order to derive statements of the form $\neg k \, says \, \varphi$, we have to assume the statements issued by a principal to be a complete characterization of what the principal supports. This is similar to the motivation behind Moore's autoepistemic logic (AEL) to consider an agent's theory to be a complete characterization of what the agent knows [Moo85b, Lev90, Nie91, DMT11]. This motivates an application of AEL to access control.

However, AEL cannot model more than one agent. In order to extend it to the multi-agent case, one needs to specify how the knowledge of the agents interacts. Most state-of-the-art access control logics allow $j \, says \, (k \, says \, \varphi)$ to be derived from $k \, says \, \varphi$, as this is required for standard delegation to be naturally modelled using the *says*-modality. In the knowledge terminology of AEL, this can be called mutual positive introspection between agents. In order to also model denial as described above, we also need mutual negative introspection, i.e. that $j \, says \, (\neg k \, says \, \varphi)$ to be derived from $\neg k \, says \, \varphi$. Van Hertum et al. [VHCBD16] have defined the semantics of dAEL(ID) in such a way that mutual positive and negative introspection between the agents is ensured.

dAEL(ID) also incorporates inductive definitions, thus allowing principals to define access rights and other properties relevant for access control in an inductive way. Inductive (recursive) definitions are a common concept in all branches of mathematics. Inductive definitions in dAEL(ID) are intended to be understood in the same way as in the general purpose specification language FO($\cdot$) of the IDP system [DBBD14b]. Denecker [Den00] showed that in classical logics, adding definitions leads to a strictly more expressive language.

Most of the semantics that have been proposed for logic programs can be adapted to inductive definitions. Denecker and Vennekens [DV14] have argued that the

well-founded semantics correctly formalizes our intuitive understanding of inductive definitions, and hence that it is actually the *right* semantics. Following them, we use the well-founded semantics for inductive definitions.

4.2 Autoepistemic Logic AEL

We introduce the basic concepts of autoepistemic logic [Moo85b]. Autoepistemic logic augments first-order logic with the modal operator for knowledge K.

Informally, the modal formula $K\varphi$ is to be read as "it is *known* that φ". Contrary to other epistemic theories, we will not distinguish between *belief* and *knowledge*, and may use these terms interchangeably.

The language of AEL is defined using the standard rules for the syntax of FO (Definition 2.3) augmented with the epistemic modality K.

Definition 4.1 (AEL-Syntax)**.** *The set of formulae of $\mathscr{L}_{AEL}$ is defined by the following EBNF rule:*

$$\varphi := P(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi \wedge \varphi \mid t = t \mid \forall x\varphi \mid K\varphi$$

The symbols $\vee, \Rightarrow, \Leftrightarrow$ and $\exists$ are treated as abbreviations in the standard way.

We will say *theory* for a conjunctively interpreted collection of formulae.

Definition 4.2 (AEL-Theory)**.** *An AEL theory T is a set consisting of AEL formulae.*

We have all the syntactic machinery in place, we now start with the semantic considerations.

In order to define the semantics for AEL we first introduce possible-world structures.

A structure, as defined in Definition 2.6, represents a state of affairs. It gives an account of a state of the world. We extend this concept to a set of structures that can represent several states of affairs that can be possible in our world.

Definition 4.3 (Possible-World Structure)**.** *A* possible-world structure (*pws*) Q *is a set of structures.*

A *pws* contains all structures that are consistent with the agents' beliefs. We can order this set of structures with respect to the amount of knowledge they contain.

Definition 4.4 (Ordering-*pws*)**.** *Given two pws* Q_1 *and* Q_2*, we define the following order over pws's:* $Q_1 \leq_K Q_2$ *if and only if* $Q_1 \supseteq Q_2$*.*

When $Q_1 \leq_K Q_2$, we say that Q_2 contains more knowledge then Q_1.

We must point out that the the knowledge ordering is inverse to the subset ordering. Intuitively in a *pws* containing less elements, some of the uncertainties about possible knowledge states (or possibilities) have been 'settled'. Having more elements reflect a higher level of uncertainty.

In the following examples, and for most part of the chapter, we will restrict ourselves to use only the propositional fragment of the logic. That is, we assume that all predicate symbols have arity 0, i.e. are *propositional variables*. Thus, the domain becomes irrelevant at this point, and we omit the domain from the structure when it is not required.

For this propositional fragment we use set notation to express the structures by listing the cases in which propositional variables are fixed to have the truth value *true*. For example, if we have a language with two propositional variables p and q, $\{p\}$ denotes the structure that makes p true and q false.

Example 4.1 (*pws*)**.** *Consider the vocabulary with only one propositional variable* p*. There are four possible world structures:*

- $Q_0 = \{\{\}, \{p\}\}$ *represents the lack of knowledge, every structure is considered possible. We write* $\bot$ *for this pws.* $\bot$ *represents the state of complete uncertainty.*

- $Q_1 = \{\{p\}\}$ *represents the knowledge of* p *being* true*.*

- $Q_2 = \{\{\}\}$ *represents the knowledge that* p *is* false.
- $Q_3 = \emptyset$ *represents the inconsistent set of beliefs. We write* $\top$ *for this pws.*

In the pws $\bot$ *we have a state of uncertainty,* p *can be either* true *or* false. *By fixing the truth value of* p *to be true we get* Q_1 *and if we fix* p *to be false we get* Q_2. *We have that* Q_1, Q_2 *contain more knowledge than* $\bot$, *that is,* $\bot \leq_K Q_1$, $\bot \leq_K Q_2$, *since* $\bot \supseteq Q_1$ *and* $\bot \supseteq Q_2$.

The complete (partial) ordering over the pws's is as follows: $\bot \leq_K Q_1$, $\bot \leq_K Q_2$, $\bot \leq_K \top$, $Q_1 \leq_K \top$, *and* $Q_2 \leq_K \top$.

We now define a two-valued valuation for the formulae of AEL with respect to a *pws* (which represents the state of mind of an agent) and a structure, representing the actual state of the world.

Definition 4.5 (Two-Valued AEL-Valuation)**.** *We inductively define a* valuation $\varphi^{Q,I} \in \{\mathbf{t}, \mathbf{f}\}$ *for an AEL formula* φ *with respect to a pws* Q *and a structure* I *as follows:*

$(P(\bar{t}))^{Q,I} = \mathbf{t}$	*iff* $t^I \in P^I$
$(t_1 = t_2)^{Q,I} = \mathbf{t}$	*iff* $t_1^I = t_2^I$
$(\varphi_1 \wedge \varphi_2)^{Q,I} = \mathbf{t}$	*iff* $(\varphi_1)^{Q,I} = \mathbf{t}$ *and* $(\varphi_2)^{Q,I} = \mathbf{t}$
$(\neg\varphi)^{Q,I} = \mathbf{t}$	*iff* $(\varphi)^{Q,I} = \mathbf{f}$
$(\forall x\ \varphi)^{Q,I} = \mathbf{t}$	*iff for each* $d \in D$, $(\varphi[x/d])^{Q,I} = \mathbf{t}$
$(K\varphi)^{Q,I} = \mathbf{t}$	*iff* $\varphi^{(Q,J)} = \mathbf{t}$ *for all* $J \in Q$

We define the two-valued valuation $T^{Q,I} \in \{\mathbf{t}, \mathbf{f}\}$ of an AEL theory T by $T^{Q,I} = \mathbf{t}$ iff $\varphi^{Q,I} = \mathbf{t}$ for all $\varphi \in T$

The notion of *supported model* is defined as follows.

Definition 4.6 (Supported Model)**.** *A pws Q is a* supported model *of a theory T iff:*

$$Q = \{I \mid T^{Q,I} = \mathbf{t}\}$$

Note that a supported model only contains those structures that make the theory true, and no other structure. That is, a supported model Q contains *exactly all* structures I such that I makes T true given Q (i.e. $T^{Q,I} = \mathbf{t}$).

We must point out that a supported model is what Moore [Moo85b] originally called the *autoepistemic expansion* of the theory.

In the following example we show how to calculate some simple supported models. Even though this minimal examples may seem trivial, thanks to the small number of *pws*'s we can iterate over the whole set of possible structures.

Example 4.2 (Supported Model Example)**.**

a) $T = \{Kp \Rightarrow p\}$*: we have the following pws's:* $\bot = \{\{p\}, \{\}\}$, $Q_1 = \{\{p\}\}$, $Q_2 = \{\{\}\}$, $\top = \emptyset$.

First we check the truth value of the propositional variables, since it does not depend on the pws's:

(i) $p^{Q,\{\}} = \mathbf{f}$ *for any pws Q; and*

(ii) $p^{Q,\{p\}} = \mathbf{t}$ *for any pws Q.*

Now calculate the valuation of the formulae contained in T for each pws:

$\bot$*:*

$(Kp)^{\bot,I} = \mathbf{f}$ *for any I, since* $p^{\bot,\{\}} = \mathbf{f}$ *and* $\{\} \in \bot$.

So, $(Kp \Rightarrow p)^{\bot,I} = \mathbf{t}$ *for any I. Therefore,* $\bot = \{I \mid T^{\bot,I} = \mathbf{t}\}$*, i.e. $\bot$ is a* supported model *for T.*

Q_1*:*

$(Kp)^{Q_1,I} = \mathbf{t}$ *since,* $Q_1 = \{\{p\}\}$ *and* $p^{Q_1,\{p\}} = \mathbf{t}$.

So, $(Kp \Rightarrow p)^{Q_1,I} = \mathbf{t}$ *iff* $I = \{p\}$*. Therefore,* $Q_1 = \{I \mid T^{Q_1,I} = \mathbf{t}\}$*, i.e.* Q_1 *is a* supported model *for* T*. Note that* $(Kp \Rightarrow p)^{Q_1,\{\}} = \mathbf{f}$*, that is,* Q_1 *contains* exactly *all* I *that make* T *true given* Q_1*.*

Q_2*:*

$(Kp)^{Q_2,I} = \mathbf{f}$ *for any* I *since,* $p^{Q_2,\{\}} = \mathbf{f}$ *and* $\{\} \in Q_2$*.*

So, $(Kp \Rightarrow p)^{Q_2,I} = \mathbf{t}$ *for any* I*. Since* $\{p\} \notin Q_2$*, it follows that* Q_2 *is not a supported model for* T*.*

$\top$*:*

$(Kp)^{\top,I} = \mathbf{t}$ *is vacuously true since there is no element in the pws* $\top$*;*

So, $(Kp \Rightarrow p)^{\top,I} = \mathbf{t}$ *iff* $I = \{p\}$*. Since* $\{p\} \notin \top$*, it follows that* $\top$ *is not a supported model for* T*.*

Thus, the supported models for T *are the pws's* $\bot$ *and* $\{\{p\}\}$*. This means that following the supported model semantics, an agent with theory* $T = \{Kp \Rightarrow p\}$ *might have no knowledge concerning the truth value of* p *or* $\neg p$*, or might know* p*, but certainly does not know* $\neg p$*.*

b) $T = \{Kp \Leftrightarrow p\}$*: we have the following pws's:* $\bot = \{\{p\}, \{\}\}$*,* $Q_1 = \{\{p\}\}$*,* $Q_2 = \{\{\}\}$*,* $\top = \emptyset$*.*

The truth assignments for the propositional variables are the same as in a): (i) $p^{Q,\{\}} = \mathbf{f}$*, and (ii)* $p^{Q,\{p\}} = \mathbf{t}$*; for any pws* Q*.*

Now we calculate the valuation of the formulae contained in T *for each pws:*

$\bot$*:*

$(Kp)^{\bot,I} = \mathbf{f}$ *for any* I *since,* $p^{\bot,\{\}} = \mathbf{f}$ *and* $\{\} \in \bot$*.*

So, $(Kp \Leftrightarrow p)^{\bot,I} = \mathbf{t}$ *iff* $I = \{\}$*. Since* $\{p\} \in \bot$*, it follows that* $\bot$ *is not a supported model.*

Q_1*:*

$(Kp)^{Q_1,I} = \mathbf{t}$ *since* $Q_1 = \{\{p\}\}$ *and* $p^{Q_1,\{p\}} = \mathbf{t}$*.*

So, $(Kp \Leftrightarrow p)^{Q_1,I} = \mathbf{t}$ *iff* $I = \{p\}$*. Therefore,* $Q_1 = \{I \mid T^{Q_1,I} = \mathbf{t}\}$*, i.e.* Q_1 *is a* supported model *for* T*.*

Q_2:

$(Kp)^{Q_2,I} = \mathbf{f}$ *for any* I *since,* $p^{Q_2,\{\}} = \mathbf{f}$ *and* $\{\} \in Q_2$,

So, $(Kp \Leftrightarrow p)^{Q_2,I} = \mathbf{t}$ *iff* $I = \{\}$. *Therefore,* $Q_2 = \{I \mid T^{Q_2,I} = \mathbf{t}\}$, *i.e.* Q_2 *is a* supported model *for* T.

$\top$:

$(Kp)^{\top,I} = \mathbf{t}$ *is vacuously true since there is no element in the pws* $\top$;

So, $(Kp \Leftrightarrow p)^{\top,I} = \mathbf{t}$ *iff* $I = \{p\}$. *Since* $\{p\} \notin \top$, *it follows that* $\top$ *is not a supported model for* T.

Thus, the supported models for T *are the pws's* $\{\{p\}\}$ *and* $\{\{\}\}$. *This means that following the supported model semantics, an agent with theory* $T = \{Kp \Leftrightarrow p\}$ *might either know* p *or might know* $\neg p$, *but certainly knows precisely one of the two.*

Supported models (more precisely autoepistemic expasions semantics) lead to certain anomalies in the constructed models. Some of these anomalies have been pointed out Halpern and Moses [HM85], and Konolige [Kon88]. Subsequently, various semantics for AEL have been proposed over time in order to avoid these anomalies. Denecker *et al.* [DMT04] showed that most proposed semantics of AEL can be defined by direct application of Approximation Fixpoint Theory (AFT). We will define four further semantics for AEL following the methodology of AFT.

Before defining these four further semantics, we first present an alternative definition of the stable semantics that motivates the semantic constructions that we will introduce thereafter. For this, we will first define an operator D_T on the lattice of *pws*'s.

Definition 4.7 (D_T Operator)**.** *Given an AEL theory* T, *we define the operator* D_T *over pws's as follows:*

$$D_T(Q) = \{I \mid T^{Q,I} = \mathbf{t}\}$$

Now we can re-define the concept of supported models in terms of the fixpoints of such operator.

A *pws* Q is a *supported model* of T if and only if $D_T(Q) = Q$. In other words, Q is a supported model of T if Q is a *fixpoint* of D_T.

We take this fixpoint construction as basis for defining different semantics for AEL using Approximation Fixpoint Theory. The idea is that we approximate the operator D_T by an *approximator* and study fixpoints of the approximator.

To approximate the operator D_T, we approximate each *pws* by an interval called belief pair. Then the approximator of D_T applies to such interval.

Definition 4.8 (Belief Pair)**.** *A* belief pair B *is a pair is a tuple* (B^c, B^l) *where* B^c *and* B^l *are two pws's.*

We define a precision ordering for belief pairs.

Definition 4.9 (Precision Ordering on Belief Pairs)**.** *Given to belief pairs* B_1 *and* B_2*, we define the following order over belief pairs:* $B_1 \leq_p B_2$ *if and only if* $B_1^c \leq_K B_2^c$ *and* $B_1^l \geq_K B_2^l$

Following the preliminary discussion in Section 2.4, we are only interested in a particular set of belief pairs.

Definition 4.10 (Consistent Belief Pair)**.** *A belief pair* B *is consistent if an only if* $B^c \leq_K B^l$ *(i.e.* $B^c \supseteq B^l$*)*

Belief pairs are used to *approximate* a *pws* Q such that $B^c \leq_K Q \leq_K B^l$. Therefore, we are only interested in consistent belief pairs where B^c is a lower bound for Q and B^l is the upper bound for Q. We call B^c the *conservative* bound and B^l the *liberal* bound for Q.

From this point forward we use only consistent belief pairs and will refer to them collectively as belief pair without any distinction.

We now define a three-valued semantic valuation. The main difference respect with the previous semantical characterization is that the boolean connectives, i.e.

$\neg$ and $\wedge$, are interpreted as defined in Kleene's three-valued logic (see Section 2.2). Additionally, a three-valued valuation is defined for the modal operator K.

We again use truth values $\mathbf{t}$ for truth and $\mathbf{f}$ for falsity, and now we have a value $\mathbf{u}$ for undefined. The truth order $<_t$ on truth values is induced by $\mathbf{f} <_t \mathbf{u} <_t \mathbf{t}$. The precision order $<_p$ on truth values is induced by $\mathbf{u} <_p \mathbf{t}$, $\mathbf{u} <_p \mathbf{f}$. We define $\mathbf{t}^{-1} = \mathbf{f}$, $\mathbf{f}^{-1} = \mathbf{t}$ and $\mathbf{u}^{-1} = \mathbf{u}$.

Definition 4.11 (Three-Valued *AEL* Valuation)**.** *We inductively define a three-valued valuation of AEL formulas with respect to a consistent belief pair B and a structure I (we write $\varphi^{B,I}$) as follows:*

$$
\begin{aligned}
(P(\bar{t}))^{B,I} &= \begin{cases} \mathbf{t} & \text{if } \bar{t}^I \in P^I \\ \mathbf{f} & \text{if } \bar{t}^I \notin P^I \end{cases} \\
(\neg\varphi)^{B,I} &= (\varphi^{B,I})^{-1} \\
(\varphi \wedge \psi)^{B,I} &= glb_{\leq_t}(\varphi^{B,I}, \psi^{B,I}) \\
(\forall x\ \varphi)^{B,I} &= glb_{\leq_t}\{\varphi[x/d]^{B,I} \mid d \in D\} \\
(K\varphi)^{B,I} &= \begin{cases} \mathbf{t} & \text{if } \varphi^{B,I'} = \mathbf{t} \text{ for all } I' \in B^c \\ \mathbf{f} & \text{if } \varphi^{B,I'} = \mathbf{f} \text{ for some } I' \in B^l \\ \mathbf{u} & \text{otherwise} \end{cases}
\end{aligned}
$$

We define the three-valued valuation $T^{B,I} \in \{\mathbf{t}, \mathbf{u}, \mathbf{f}\}$ of an *AEL* theory T by $T^{B,I} = glb_{\leq_t}(\{\varphi^{B,I} \mid \varphi \in T\})$

We approximate the value of the operator D_T by defining the approximator D_T^*, an operator on belief pairs [DMT00].

Definition 4.12 (D_T^* Approximator)**.** *We define the approximator D_T^* over consistent belief pairs as follows:*

$$D_T^*(B) = (\{I \mid T^{B,I} \neq \mathbf{f}\}, \{I \mid T^{B,I} = \mathbf{t}\})$$

Intuitively, the approximator D^*_T gathers: *(i)* all structures I such that I makes T *true* or *undecided* given B, for the conservative bound; and, *(ii)* all structures I such that I makes T *true* given B, for the liberal bound.

The approximator D^*_T maps a belief pair B to a belief pair B'. The new conservative bound B'^c contains all the knowledge that can be *certainly* derived from the current state. The new liberal bound B'^l contains all the knowledge that can be *possibly* derived from the current state.

By the use of approximators instead of just a simple fixpoint characterization (the operator D_T) a class of semantics emerge for AEL. Namely the following semantics can be defined:

- Kripke-Kleene semantics [DMT98] (Kripke-Kleene fixpoints).
- Stable semantics [DMT03] (stable fixpoints).
- Partial Stable semantics [DMT03] (partial stable fixpoints).
- Well-founded semantics [DMT03] (well-founded fixpoints).

Well-founded and stable semantics were new semantics induced by the usage of AFT. These semantics closely correspond to the same semantic defined for logic programs. We are especially interested in the well-founded semantics, since this semantic—afterwards extended to the multi-agent case—has the desired behaviour for access control applications [VHCBD16].

As discussed in Section 2.4, AFT in general presupposes an approximator defined for the whole bi-lattice L^2, i.e. also for inconsistent pairs. However, for a symmetric operator A on L^2, the restriction A^c of A to L^c has the same consistent fixpoints of the different kinds (KK, Stable, WF) as A [DMT04, Theorem 4.2]. So, it is enough to define an operator on L^c (as we have done) and assume that it gets extended to some symmetric operator.

We are interested only in consistent belief pairs. Nonetheless inconsistent belief pairs may appear when we iteratively calculate the fixpoints of an operator, or more precisely of an approximator.

We extend the approximator to a particular set of inconsistent belief pairs i.e. those inconsistent belief pairs that are symmetric with respect to the belief pairs in L^c. We define L' to be $L^c \cup \{(x, y) \mid (y, x) \in L^c\}$. Alternatively L' can be defined as $L' = L^c \cup (L^c)^{-1}$. We have that $L^c \subseteq L' \subseteq L$.

We extend D^*_T to L' by defining:

$$D^*_T(B^c, B^l) = (D^*_T(B^l, B^c)^l, D^*_T(B^l, B^c)^c) \text{ whenever } B^l \leq_K B^c$$

The importance of this definition can be seen in the coming examples.

Definition 4.13 (Kripke-Kleene Model)**.** *The belief pair B is the* Kripke-Kleene model *of T iff B is the least precise fixpoint of D^*_T.*

The *Kripke-Kleene* model always exists, since the operator D^*_T is monotone [DMT03, Prop 3.9]. When the domain is finite its construction is decidable. For a finite theory T over a finite domain, the Kripke-Kleene model can be computed using the following algorithm:

Algorithm 1 Kripke-Kleene Model

Require: finite theory T over a finite domain
Ensure: the KK-Model for T
1: Let $B_0 := (\bot, \top)$
2: **repeat**
3: $\quad B_{i+1} := D^*_T(B_i)$
4: **until** $D^*_T(B_k) = B_k$
5: **return** B_k

Proposition 1 (Algorithm 1 Correctness)**.** *Let T be a finite AEL theory, where the domain also finite. Algorithm 1 returns the Krike-Kleene model for T.*

Proof. Follows directly from the fact that the operator D^*_T is a monotone operator [DMT03, Prop 3.9]. This ensures the existence of a fixpoint for the operator. Since the domain (and theory) are finite, this fixpoint can be computed in finitely may steps, so the termination of the algorithm is ensured. Every iteration generates a belief pair more precise than the previous, until the least precise fixpoint is reached, that is the Kripke-Kleene model for the theory. □

In the following example we show how to apply Algorithm 1 to compute the Kripke-Kleene model for the theories in Example 4.2.

Example 4.3 (Kripke-Kleene Model Example)**.**

a) $T = \{Kp \Rightarrow p\}$*: we start with the belief pair* $B_0 := (\bot, \top)$*.*

The truth assignment for the propositional variables is as follows: (i) $p^{Q,\{\}} = \mathbf{f}$ *for any* Q*, and* (ii) $p^{Q,\{p\}} = \mathbf{t}$ *for any* Q*. From (ii) it follows that* $(Kp \Rightarrow p)^{Q,\{p\}} = \mathbf{t}$ *for any* Q*.*

We calculate B_1*:*

$(Kp)^{(\bot,\top),\{\}} = \mathbf{u}$ *since* $\exists I \in \bot$ *such that* $p^{Q,I} = \mathbf{f}$*; and* $\nexists I \in \top$ *such that* $p^{Q,I} = \mathbf{f}$*. So,* $(Kp \Rightarrow p)^{(\bot,\top),\{\}} = \mathbf{u}$ *since* $p^{Q,\{\}} = \mathbf{f}$

$$\begin{aligned} B_1 &:= D^*_T((\bot, \top)) \\ &:= (\{I \mid T^{(\bot,\top),I} \neq \mathbf{f}\}, \{I \mid T^{(\bot,\top),I} = \mathbf{t}\}) \\ &:= (\bot, \{\{p\}\}) \end{aligned}$$

We calculate B_2*:*

$(Kp)^{(\bot,\{\{p\}\}),\{\}} = \mathbf{u}$ *since* $\exists I \in \bot$ *such that* $p^{Q,I} = \mathbf{f}$ *and* $\nexists I \in \{\{p\}\}$ *such that* $p^{Q,I} = \mathbf{f}$*. So,* $(Kp \Rightarrow p)^{(\bot,\{\{p\}\}),\{\}} = \mathbf{u}$ *since* $p^{Q,\{\}} = \mathbf{f}$*.*

$$\begin{aligned} B_2 &:= D^*_T((\bot, \{\{p\}\})) \\ &:= (\{I \mid T^{(\bot,\{\{p\}\}),I} \neq \mathbf{f}\}, \{I \mid T^{(\bot,\{\{p\}\}),I} = \mathbf{t}\}) \\ &:= (\bot, \{\{p\}\}) \end{aligned}$$

We have reached a fixpoint, because $B_2 = B_1$*. So the Kripke-Kleene model for the theory is:* $(\bot, \{\{p\}\})$*. This means that following the Kripke-Kleene*

semantics, an agent with theory $T = \{Kp \Rightarrow p\}$ *might have no knowledge concerning* p *or* $\neg p$*, or might know* p*, but certainly does not know* $\neg p$*.*

b) $T = \{Kp \Leftrightarrow p\}$*: we start with the belief pair* $B_0 := (\bot, \top)$*.*

The truth assignment for the propositional variables is as follows:
(i) $p^{Q,\{\}} = \mathbf{f}$ *for any* Q*, and* (ii) $p^{Q,\{p\}} = \mathbf{t}$ *for any* Q

We calculate B_1*:*

$(Kp)^{(\bot,\top),\{\}} = \mathbf{u}$ *since* $\exists I \in \bot$ *such that* $p^{Q,I} = \mathbf{f}$ *and* $\nexists I \in \top$ *such that* $p^{Q,I} = \mathbf{f}$*. So,* $(Kp \Leftrightarrow p)^{(\bot,\top),\{\}} = \mathbf{u}$*.*

$(Kp)^{(\bot,\top),\{p\}} = \mathbf{u}$ *since* $\exists I \in \bot$ *such that* $p^{Q,I} = \mathbf{f}$ *and* $\nexists I \in \bot$ *such that* $p^{Q,I} = \mathbf{f}$*. So,* $(Kp \Leftrightarrow p)^{(\bot,\top),\{p\}} = \mathbf{u}$*.*

Finally, $B_1 := (\{\{\}, \{p\}\}, \emptyset) = (\bot, \top)$*.*

We have reached a fixpoint, because $B_1 = B_0$ *So the Kripke-Kleene model for the theory is:* $(\bot, \top)$*. This means that following the Kripke-Kleene semantics, an agent with theory* $T = \{Kp \Leftrightarrow p\}$ *might have no knowledge concerning* p *or* $\neg p$*, or might have the inconsistent knowledge that* p *and* $\neg p$ *both hold, or might have some intermediate knowledge, like just the knowledge of* p *or just the knowledge of* $\neg p$*.*

We procede to define the rest of the semantics described above. For that purpose we introduce a new operator, the *stable* operator D_T^{st}.

Definition 4.14 (D_T^{st} Operator)**.** *The stable operator* D_T^{st} *is the least fixpoint of the conservative bound given a fixed liberal bound, that is,* $D_T^{st}(Q) = \texttt{lfp}(D_T^*(\cdot, Q)^c)$

Intuitively, the operator D_T^{st} can be viewed as new conservative bound of what is believed given a fixed liberal point of view.

The stable operator D_T^{st} over a finite domain can be calculated using the following algorithm:

Algorithm 2 Stable Operator D_T^{st}

Require: theory T, *pws* Q
Ensure: a stable model for T
1: Let $Q_0 := \bot$
2: **repeat**
3: $\quad Q_{i+1} := D_T^*(Q_i, Q)^c$
4: **until** $Q_i = Q_{i+1}$
5: **return** Q_i

Proposition 2 (Algorithm Correctness)**.** *Let T be an AEL theory. Algorithm 2 returns $D_T^{st}(Q)$ for input T, Q.*

Proof. The proof can be constructed in a similar manner as in Proposition 1. □

Definition 4.15 (Stable Model)**.** *Let T be an AEL theory. A* stable model *of T is a pws Q, that is a fixpoint of D_T^{st}.*

There are several equivalent ways in which a *stable* and *partial stable models* can be defined, see Van Hertum *et al.* [VHCBD16] and Denecker *et al.* [DMT04] for an alternative definition.

Definition 4.16 (D_T^{st*} Approximator)**.** *We define the approximator D_T^{st*} over a belief pair B as follows:*

$$D_T^{st*} = (D_T^{st}(B^l), D_T^{st}(B^c))$$

This last approximator yields the last remaining semantical characterizations, *partial stable models* and *well-founded models*.

Definition 4.17 (Partial Stable Model)**.** *Let T be an AEL theory. A* partial stable model *of T is a fixpoint of D_T^{st*}.*

Definition 4.18 (Well-founded Model)**.** *Let T be an AEL theory. The* well-founded model (*wfm*) *of T is the least precise fixpoint of D_T^{st*}.*

The *wfm* for a finite *AEL* theory (over a finite domain) can be calculated using the following algorithm:

Algorithm 3 Well-Founded Model

Require: theory T
Ensure: the well-founded model for T
1: Let $B_0 := (\bot, \top)$
2: **repeat**
3: $\quad B^c_{i+1} := D^{st}_T(B^l_i)$ **Algorithm 2
4: $\quad B^l_{i+1} := D^{st}_T(B^c_i)$ ** Algorithm 2
5: **until** $B_{i+1} = B_i$
6: **return** B_i

Proposition 3 (Algorithm Correctness)**.** *Let T be an AEL theory. For input T, Algorithm 3 returns the well founded model of T.*

Proof. Follows from Proposition 2. □

Example 4.4 (Stable Model Example)**.**

a) $T = \{Kp \Rightarrow p\}$*:*

We calculate the fixpoints of D^{st}_T *by calculating* $D^{st}_T(Q)$ *for each possible pws:*

We calculate $D^{st}_T(\bot)$*:*

$Q_0 := \bot$

$(Kp)^{(\bot,\bot),\{\}} = \mathbf{f}$ *since* $\exists I \in \bot$ *such that* $p^{(\bot,\bot),I} = \mathbf{f}$*. So,* $(Kp \Rightarrow p)^{(\bot,\bot),\{\}} = \mathbf{t}$*. Similarly,* $(Kp \Rightarrow p)^{(\bot,\bot),\{p\}} = \mathbf{t}$.

$$\begin{aligned} Q_1 &:= D^*_T((\bot,\bot))^c \\ &:= (\{I \mid T^{(\bot,\bot),I} \neq \mathbf{f}\}, \{I \mid T^{(\bot,\bot),I} = \mathbf{t}\})^c \\ &:= (\bot,\bot)^c \\ &:= \bot \end{aligned}$$

We have reached a fixpoint, because $Q_1 = Q_0$*. Thus,* $D^{st}_T(\bot) = \bot$

We calculate $D^{st}_T(\{\{p\}\})$*:*

$Q_0 := \bot$

$(Kp)^{(\bot,\{\{p\}\}),\{\}} = \mathbf{u}$ *since* $\exists I \in \bot$ *such that* $p^{(\bot,\{\{p\}\}),I} = \mathbf{f}$ *and* $\nexists I \in \{\{p\}\}$ *such that* $p^{(\bot,\{\{p\}\}),I} = \mathbf{f}$. *So,* $(Kp \Rightarrow p)^{(\bot,\{\{p\}\}),\{\}} = \mathbf{u}$ *since* $p^{(\bot,\{\{p\}\}),\{\}} = \mathbf{f}$. *Similarly,* $(Kp \Rightarrow p)^{(\bot,\{\{p\}\}),\{p\}} = \mathbf{t}$ *since* $p^{(\bot,\{\{p\}\}),\{p\}} = \mathbf{t}$.

$$\begin{aligned} Q_1 &:= D^*_T((\bot,\{\{p\}\}))^c \\ &:= (\{I \mid T^{(\bot,\{\{p\}\}),I} \neq \mathbf{f}\}, \{I \mid T^{(\bot,\{\{p\}\}),I} = \mathbf{t}\})^c \\ &:= (\bot,\{\{p\}\})^c \\ &:= \bot \end{aligned}$$

We have reached a fixpoint, because $Q_1 = Q_0$. *Thus,* $D^{st}_T(\{\{p\}\}) = \bot$.

We calculate $D^{st}_T(\{\{\}\})$*:*

$Q_0 := \bot$

$(Kp)^{(\bot,\{\{\}\}),\{\}} = \mathbf{f}$ *since* $\exists I \in \{\{\}\}$ *such that* $p^{(\bot,\{\{\}\}),I} = \mathbf{f}$. *So,* $(Kp \Rightarrow p)^{(\bot,\{\{\}\}),\{\}} = \mathbf{t}$. *Similarly,* $(Kp \Rightarrow p)^{(\bot,\{\{\}\}),\{p\}} = \mathbf{t}$.

$$\begin{aligned} Q_1 &:= D^*_T((\bot,\{\{\}\}))^c \\ &:= (\bot,\bot)^c \\ &:= \bot \end{aligned}$$

We have reached a fixpoint, because $Q_1 = Q_0$. *Thus,* $D^{st}_T(\{\{\}\}) = \bot$.

We calculate $D^{st}_T(\top)$*:*

$Q_0 = \bot$

$(Kp)^{(\bot,\top),\{\}} = \mathbf{u}$ *since* $\exists I \in \bot$ *such that* $p^{(\bot,\top),I} = \mathbf{f}$ *and* $\nexists I \in \top$ *such that* $p^{(\bot,\top),I} = \mathbf{f}$. *So,* $(Kp \Rightarrow p)^{(\bot,\top),\{\}} = \mathbf{u}$ *since* $p^{(\bot,\top),\{\}} = \mathbf{f}$. *Similarly,* $(Kp \Rightarrow p)^{(\bot,\top),\{p\}} = \mathbf{t}$ *since* $p^{(\bot,\top),\{p\}} = \mathbf{t}$.

$$\begin{aligned} Q_1 &:= D^*_T((\bot,\top))^c \\ &:= (\bot,\{\{p\}\})^c \\ &:= \bot \end{aligned}$$

We have reached a fixpoint, because $Q_1 = Q_0$. *Thus,* $D^{st}_T(\top) = \bot$.

The above results imply that D_T^{st} has only one fixpoint $D_T^{st}(\bot) = \bot$. So the stable model for the theory is: $\bot$. This means that following the stable semantics, an agent with theory $T = \{Kp \Rightarrow p\}$ has no knowledge concerning the truth value of p or $\neg p$.

b) $T = \{Kp \Leftrightarrow p\}$*:*

We calculate the fixpoints of D_T^{st} by calculating $D_T^{st}(Q)$ for each possible pws:

We calculate $D_T^{st}(\bot)$:

$Q_0 := \bot$

$(Kp)^{(\bot,\bot),\{\}} = \mathbf{f}$ *since* $\exists I \in \bot$ *such that* $p^{(\bot,\bot),I} = \mathbf{f}$. *So,* $(Kp \Leftrightarrow p)^{(\bot,\bot),\{\}} = \mathbf{t}$ *since* $p^{(\bot,\bot),\{\}} = \mathbf{f}$. *Similarly,* $(Kp \Leftrightarrow p)^{(\bot,\bot),\{p\}} = \mathbf{f}$ *since* $p^{(\bot,\bot),\{p\}} = \mathbf{t}$.

$$\begin{aligned} Q_1 &:= D_T^*((\bot,\bot))^c \\ &:= (\{\{\}\},\{\{\}\})^c \\ &:= \{\{\}\} \end{aligned}$$

$Q_2 := D_T^*((\{\{\}\},\bot))^c$. *The belief pair* $(\{\{\}\},\bot)$ *is not consistent. As previously discussed, since the approximator D_T^* is symmetric we swap the belief pair to regain consistency and calculate accordingly for the new consistent belief pair.*

$(Kp)^{(\bot,\{\{\}\}),\{\}} = \mathbf{f}$ *since* $\exists I \in \{\{\}\}$ *such that* $p^{(\bot,\{\{\}\}),I} = \mathbf{f}$. *So,* $(Kp \Leftrightarrow p)^{(\bot,\{\{\}\}),\{\}} = \mathbf{t}$ *since* $p^{(\bot,\{\{\}\}),\{\}} = \mathbf{f}$. *Similarly,* $(Kp \Leftrightarrow p)^{(\bot,\{\{\}\}),\{\}} = \mathbf{f}$ *since* $p^{Q,\{p\}} = \mathbf{t}$.

$$\begin{aligned} Q_2 &:= D_T^*((\bot,\{\{\}\}))^c \\ &:= (\{\{\}\},\{\{\}\})^c \\ &:= \{\{\}\} \end{aligned}$$

We have reached a fixpoint, because $Q_2 = Q_1$. Thus, $D_T^{st}(\bot) = \bot$.

We calculate $D_T^{st}(\{\{p\}\})$:

$Q_0 := \bot$

$(Kp)^{(\bot,\{\{p\}\}),\{\}} = \mathbf{u}$ *since* $\exists I \in \bot$ *such that* $p^{(\bot,\{\{p\}\}),I} = \mathbf{f}$ *and* $\nexists I \in \{\{p\}\}$ *such that* $p^{(\bot,\{\{p\}\}),I} = \mathbf{f}$. *So,* $(Kp \Leftrightarrow p)^{(\bot,\{\{p\}\}),\{\}} = \mathbf{u}$. *Similarly,* $(Kp \Leftrightarrow p)^{(\bot,\{\{p\}\}),\{p\}} = \mathbf{u}$.

$$\begin{aligned} Q_1 &:= D_T^*((\bot, \{\{p\}\}))^c \\ &:= (\bot, \top)^c \\ &:= \bot \end{aligned}$$

We have reached a fixpoint, because $Q_1 = Q_0$. *Thus,* $D_T^{st}(\{\{p\}\}) = \bot$.

We calculate $D_T^{st}(\{\{\}\})$*:*

$Q_0 := \bot$

$(Kp)^{(\bot,\{\{\}\}),\{\}} = \mathbf{f}$ *since* $\exists I \in \{\{\}\}$ *such that* $p^{(\bot,\{\{\}\}),I} = \mathbf{f}$. *So,* $(Kp \Leftrightarrow p)^{(\bot,\{\{\}\}),\{\}} = \mathbf{t}$. *Similarly,* $(Kp \Leftrightarrow p)^{(\bot,\{\{\}\}),\{p\}} = \mathbf{f}$.

$$\begin{aligned} Q_1 &:= D_T^*((\bot, \{\{\}\}))^c \\ &:= (\{\{\}\}, \{\{\}\})^c \\ &:= \{\{\}\} \end{aligned}$$

$(Kp)^{(\{\{\}\},\{\{\}\}),\{\}} = \mathbf{f}$ *since* $\exists I \in \{\{\}\}$ *such that* $p^{(\{\{\}\},\{\{\}\}),I} = \mathbf{f}$. *So,* $(Kp \Leftrightarrow p)^{(\{\{\}\},\{\{\}\}),\{\}} = \mathbf{t}$. *Similarly,* $(Kp \Leftrightarrow p)^{(\{\{\}\},\{\{\}\}),\{p\}} = \mathbf{f}$.

$$\begin{aligned} Q_2 &:= D_T^*((\{\{\}\}, \{\{\}\}))^c \\ &:= (\{\{\}\}, \{\{\}\})^c \\ &:= \{\{\}\} \end{aligned}$$

We have reached a fixpoint, because $Q_2 = Q_1$. *Thus,* $D_T^{st}(\{\{\}\}) = \{\{\}\}$

We calculate $D_T^{st}(\top)$*:*

$Q_0 = \bot$

$(Kp)^{(\bot,\top),\{\}} = \mathbf{u}$ *since* $\exists I \in \bot$ *such that* $p^{(\bot,\top),I} = \mathbf{f}$ *and* $\nexists I \in \top$ *such that* $p^{(\bot,\top),I} = \mathbf{f}$. *So,* $(Kp \Leftrightarrow p)^{(\bot,\top),\{\}} = \mathbf{u}$. *Similarly,* $(Kp \Leftrightarrow p)^{(\bot,\top),\{p\}} = \mathbf{u}$.

$$\begin{aligned} Q_1 &:= D_T^*((\bot, \top))^c \\ &:= (\bot, \top)^c \\ &:= \bot \end{aligned}$$

We have reached a fixpoint, because $Q_1 = Q_0$. *Thus,* $D_T^{st}(\top) = \bot$.

The above results mean that D_T^{st} *has only one fixpoint:* $D_T^{st}(\{\{\}\}) = \{\{\}\}$. *So the stable model for the theory is:* $\{\{\}\}$. *This means that following the stable semantics, an agent with theory* $T = \{Kp \Leftrightarrow p\}$ *certainly knows* $\neg p$.

The following example shows how to calculate the well-founded model. We use the results from Example 4.4.

Example 4.5 (Well-Founded Model Example)**.**

a) $T = \{Kp \Rightarrow p\}$*: we start with the belief pair* $B_0 := (\bot, \top)$.

We calculate B_1*:*

$B_1^c := D_T^{st}(\top) = \bot$

$B_1^l := D_T^{st}(\bot) = \bot$

We have $B_1 := (\bot, \bot)$.

We calculate B_2*:*

$B_2^c := D_T^{st}(\bot) = \bot$

$B_2^l := D_T^{st}(\bot) = \bot$

We have $B_2 := (\bot, \bot)$.

We have reached a fixpoint, because $B_2 = B_1$. *So the Well-founded model for the theory is:* $(\bot, \bot)$. *This means that following the well-founded semantics, an agent with theory* $T = \{Kp \Rightarrow p\}$ *has no knowledge concerning the truth value of* p *or* $\neg p$.

b) $T = \{Kp \Leftrightarrow p\}$*: we start with the belief pair* $B_0 := (\bot, \top)$.

We calculate B_1*:*

$B_1^c := D_T^{st}(\top) = \bot$

$B_1^l := D_T^{st}(\bot) = \{\{\}\}$

We have $B_1 := (\bot, \{\{\}\})$.

We calculate B_2*:*

$B_2^c := D_T^{st}(\{\{\}\}) = \{\{\}\}$

$B_2^l := D_T^{st}(\bot) = \{\{\}\}$

We have $B_2 := (\{\{\}\}, \{\{\}\})$.

We calculate B_3*:*

$B_3^c := D_T^{st}(\{\{\}\}) = \{\{\}\}$

$B_3^l := D_T^{st}(\{\{\}\}) = \{\{\}\}$

We have $B_3 := (\{\{\}\}, \{\{\}\})$.

We have reached a fixpoint, because $B_3 = B_2$ *So the Well-founded model for the theory is:* $(\{\{\}\}, \{\{\}\})$*. This means that following the well-founded semantics, an agent with theory* $T = \{Kp \Leftrightarrow p\}$ *certainly knows that* $\neg p$*.*

4.3 Distributed Autoepistemic Logic - *dAEL*

Now we introduce a multi-agent variant for *AEL* called *dAEL* first introduced by Van Hertum *et al.* [VHCBD16].

Throughout the remainder of the chapter, we assume a set $\mathcal{A}$ of *principals* and a first-order vocabulary Σ with a finite domain, they are both assumed given and fixed (except stated otherwise).

Definition 4.19 (*dAEL* Syntax)**.** *dAEL formulas are defined by the following EBNF rule, where* t *denotes an arbitrary term and* x *and arbitrary variable:*

$$\varphi ::= P(t, \ldots, t) \mid t = t \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \forall x\ \varphi \mid t \ \ says \ \ \varphi$$

The symbols $\vee$, $\Rightarrow$, $\Leftrightarrow$ and $\exists$ are treated as abbreviations in the standard way.

The intuitive reading of t *says* φ is "t is a principal and t supports φ". So if the term t does not denote a principal, t *says* φ will be interpreted to be false.

In [VHCBD16], *t says* φ is also written as $K_t\varphi$, which is more in line with the notation of standard autoepistemic logic. We chose the former to be in line with the notation in the access control literature. We have motivated the usage of autoepistemic logic as an access control logic.

Inductive definitions are generally used to define a reduced set of predicates. We call these *defined predicates*. The rest of the predicates defined in the language (i.e. outside the inductive definition) are called *parameters*.

Definition 4.20. *Let* $\Delta = \{P_1(\bar{t}_1) \leftarrow \varphi_1, \ldots, P_n(\bar{t}_n) \leftarrow \varphi_n\}$ *be an inductive definition. Then Def*(Δ) *is defined to be* $\{P_1, \ldots, P_n\}$ *and is called the set of* defined predicates *of* Δ*. The set of predicates in* Σ *that are not in Def*(Δ) *is denoted Par*(Δ) *and is called the set of* parameters *of* Δ*.*

Definition 4.21 ($dAEL$-Theory)**.** *A* $dAEL$-Theory *is a set that consists of* $dAEL$ *formulas and inductive definitions.*

Definition 4.22 (Distributed Theory)**.** *A* distributed theory *is an indexed family* $\mathcal{T} = (\mathcal{T}_A)_{A \in \mathcal{A}}$ *where each* $\mathcal{T}_A$ *is a* $dAEL$*-Theory.*

We now start with the semantical considerations. We lift all the common definitions already introduced for the semantics of AEL to the distributed case in $dAEL$.

A *pws* (as in Definition 4.3) represents the possible states of affairs for a particular theory, i.e. one principal. We extend this concept, to a set of *pws*'s for each principal on the system.

Definition 4.23 (Distributed PWS)**.** *A* distributed possible-world structure (*dpws*) $\mathcal{Q} = (\mathcal{Q}_A)_{A \in \mathcal{A}}$ *is a family of pws's* Q_A *for each principal* $A \in \mathcal{A}$*.*

We define again a precedence ordering, now over *dpws*.

Definition 4.24 (Precedence Ordering on *dpws*)**.** *Given two dpws* $\mathcal{Q}^1$ *and* $\mathcal{Q}^2$*, we define the following order over dpws's:* $\mathcal{Q}^1 \leq_K \mathcal{Q}^2$ *if and only if* $\mathcal{Q}^1_A \leq_K \mathcal{Q}^2_A$ *for each* $A \in \mathcal{A}$*.*

We now define a two-valued valuation for the formulae of $dAEL$ with respect to *dpws*, which represents the collective state of mind of the agents, and a structure representing the actual state of the world.

Definition 4.25 (Two-Valued $dAEL$ Valuation)**.** *We inductively define a* two-valued valuation *of dAEL(ID) formulas with respect to a dpws $\mathcal{Q}$ and a structure I (we write $\varphi^{\mathcal{Q},I}$) as follows:*

$$
\begin{array}{ll}
(P(\bar{t}))^{\mathcal{Q},I} = \mathbf{t} & \textit{iff } \bar{t}^I \in P^I \\
(t_1 = t_2)^{\mathcal{Q},I} = \mathbf{t} & \textit{iff } t_1^I = t_2^I \\
(\varphi_1 \wedge \varphi_2)^{\mathcal{Q},I} = \mathbf{t} & \textit{iff } (\varphi_1)^{\mathcal{Q},I} = \mathbf{t} \textit{ and } (\varphi_2)^{\mathcal{Q},I} = \mathbf{t} \\
(\neg\varphi)^{\mathcal{Q},I} = \mathbf{t} & \textit{iff } (\varphi)^{\mathcal{Q},I} = \mathbf{f} \\
(\forall x\ \varphi)^{\mathcal{Q},I} = \mathbf{t} & \textit{iff for each } d \in D,\ (\varphi[x/d])^{\mathcal{Q},I} = \mathbf{t} \\
(t \textit{ says } \varphi)^{\mathcal{Q},I} = \mathbf{t} & \textit{iff } t^I \in \mathcal{A} \textit{ and } \varphi^{(\mathcal{Q},J)} = \mathbf{t} \textit{ for all } J \in \mathcal{Q}_{t^I}
\end{array}
$$

We define the two-valued valuation $T^{\mathcal{Q},I} \in \{\mathbf{t}, \mathbf{f}\}$ of a $dAEL$ theory T by $T^{\mathcal{Q},I} = \mathbf{t}$ iff $\varphi^{\mathcal{Q},I} = \mathbf{t}$ for all $\varphi \in T$.

The set of *dpws*'s equipped with the precision ordering forms a complete lattice. We define an operator $D_{\mathcal{T}}$ over this lattice.

Definition 4.26 ($D_{\mathcal{T}}$ Operator)**.** *We define the operator $D_{\mathcal{T}}$ over dpws's as follows:*

$$D_{\mathcal{T}}(\mathcal{Q}) = (\{I \mid (\mathcal{T}_A)^{\mathcal{Q},I} = \mathbf{t}\})_{A \in \mathcal{A}}$$

To approximate the operator $D_{\mathcal{T}}$, we build an interval called distributed belief pair. Intuitively, a belief pair represents an approximation of the state of mind of the the set of agents. The pair consists of a conservative bound B^c and a liberal bound B^l of each agent's state of mind.

Definition 4.27 (Distributed Belief Pair)**.** *A* distributed belief pair $\mathcal{B}$ *is a pair $\mathcal{B} = (\mathcal{B}^c, \mathcal{B}^l)$ of two dpws's $\mathcal{B}^c$, and $\mathcal{B}^l$ such that $\mathcal{B}^c \leq_K \mathcal{B}^l$ of dpws's.*

We define a precision ordering for distributed belief pairs.

Definition 4.28 (Precision Ordering on Distributed Belief-Pairs)**.** *Given to belief pairs* $\mathcal{B}_1$ *and* $\mathcal{B}_2$*, we define the following order over belief pairs:* $\mathcal{B}_1 \leq_p \mathcal{B}_2$ *if and only if* $\mathcal{B}_1^c \leq_K \mathcal{B}_2^c$ *and* $\mathcal{B}_1^l \geq_K \mathcal{B}_2^l$

We now define a two-valued valuation for the formulae of *dAEL* with respect to distributed belief pair and a structure representing the actual state of the world.

Definition 4.29 (Three-Valued *dAEL* Valuation)**.** *We inductively define a* three-valued valuation *of dAEL(ID) formulas with respect to a distributed belief pair* $\mathcal{B}$ *and a structure* I *as follows:*

$$
\begin{aligned}
(P(\bar{t}))^{\mathcal{B},I} = & \quad \begin{cases} \mathbf{t} & \text{if } \bar{t}^I \in P^I \\ \mathbf{f} & \text{if } \bar{t}^I \notin P^I \end{cases} \\
(\neg\varphi)^{\mathcal{B},I} = & \quad (\varphi^{\mathcal{B},I})^{-1} \\
(\varphi \wedge \psi)^{\mathcal{B},I} = & \quad glb_{\leq_t}(\varphi^{\mathcal{B},I}, \psi^{\mathcal{B},I}) \\
(\forall x\ \varphi)^{\mathcal{B},I} = & \quad glb_{\leq_t}\{\varphi[x/d]^{\mathcal{B},I} \mid d \in D\} \\
(t \text{ says } \varphi)^{\mathcal{B},I} = & \quad \begin{cases} \mathbf{t} & \text{if } t^I \in \mathcal{A} \text{ and} \\ & \quad \varphi^{\mathcal{B},I'} = \mathbf{t} \text{ for all } I' \in \mathcal{B}^c_{t^I} \\ \mathbf{f} & \text{if } t^I \notin \mathcal{A} \text{ or} \\ & \quad \varphi^{\mathcal{B},I'} = \mathbf{f} \text{ for some } I' \in \mathcal{B}^l_{t^I} \\ \mathbf{u} & \text{otherwise} \end{cases}
\end{aligned}
$$

We define the three-valued valuation $T^{\mathcal{B},I} \in \{\mathbf{t}, \mathbf{u}, \mathbf{f}\}$ of a *dAEL* theory T by $T^{\mathcal{B},I} = glb_{\leq_t}(\{\varphi^{\mathcal{B},I} \mid \varphi \in T\})$

Inductive definitions in dAEL(ID) are interpreted according to the well-founded semantics for inductive definitions, for details see [DV14]. The well-founded model of an inductive definition Δ is always defined relative to a context $\mathcal{O}$. This context is an interpretation of the predicate symbols outside the inductive definition (i.e. $Par(\Delta)$). We denote the well-founded model of Δ relative to $\mathcal{O}$ by $wfm_\Delta(\mathcal{O})$.

Inductive definitions in dAEL(ID) may contain the *says*-modality in the body of the definition. The well-founded model in [DV14] is only defined for inductive definition over a first-order language without any modality. We define how to interpret the *says*-modality in the body. We evaluate inductive definitions with respect to a *dpws* $\mathcal{Q}$ and a structure I. The *dpws* $\mathcal{Q}$ assigns a truth-value to every formula of the form $k\ says\ \varphi$. When evaluating an inductive definition Δ with respect to $\mathcal{Q}$ and I, it should get evaluated in the same way as the inductive definition $\Delta^{\mathcal{Q}}$, which is defined to be Δ with all instances of formulas of the form $k\ says\ \varphi$ replaced by $\mathbf{t}$ or $\mathbf{f}$ according to their interpretation in $\mathcal{Q}$.

As already stated in the motivation, we are interested in the well-founded model for the theories with respect to our application on the access control logic domain. Thus we define a three-valued valuation of dAEL(ID) inductive definitions with respect to a distributed belief pair $\mathcal{B}$ and a structure I in terms of well-founded semantics for dAEL(ID).

Definition 4.30 (Three-Valued dAEL(ID) Inductive Definition Valuation)**.** *We define a three-valued valuation of dAEL(ID) inductive definitions with respect to a distributed belief pair $\mathcal{B}$ and a structure I as follows:*

$$\Delta^{\mathcal{B},I} = \begin{cases} \mathbf{t} & \text{iff } I = wfm_{\Delta^{\mathcal{B}}}(I|_{Par(\Delta)}) \\ \mathbf{f} & \text{iff } I \not\geq_p wfm_{\Delta^{\mathcal{B}}}(I|_{Par(\Delta)}) \\ \mathbf{u} & \text{otherwise} \end{cases}$$

where $\Delta^{\mathcal{B}}$ is the definition Δ with all formulas $t\ says\ \varphi$ replaced by $\mathbf{t}$, $\mathbf{f}$ or $\mathbf{u}$, according to their interpretation in $\mathcal{B}$.

In a partial context ($\mathcal{B}$ is three-valued), we cannot yet evaluate the exact value of the predicates in $Def(\Delta)$. We can, however, obtain an approximation $wfm_{\Delta^{\mathcal{B}}}(I|_{Par(\Delta)})$ of their value. Thus, the three-valued valuation of dAEL(ID) inductive definitions can be intuitively understood as follows: *(i)* $\Delta^{\mathcal{B},I} = \mathbf{t}$ if the approximation is actually two-valued and equal to I, *(ii)* $\Delta^{\mathcal{B},I} = \mathbf{u}$ if I is still

consistent with–but not equal to–this approximation, and *(iii)* $\Delta^{\mathcal{B},I} = \mathbf{f}$ in any other case.

We can combine the three-valued valuations for formulas and inductive definitions into a three-valued valuation of a single agent's theory as follows:

Definition 4.31 (Three-Valued dAEL(ID) Theory Valuation)**.** *We define a three-valued valuation of dAEL(ID) theories with respect to a distributed belief pair $\mathcal{B}$ and a structure I as follows:*

$$T^{\mathcal{B},I} := glb_{\leq_t}(\{\varphi^{\mathcal{B},I} | \varphi \in \mathcal{T}_A\} \cup \{\Delta^{\mathcal{B},I} | \Delta \in \mathcal{T}_A\})$$

We define the operators and approximators required to define the well-founded semantics for dAEL(ID).

Definition 4.32 ($D^*_{\mathcal{T}}$ Approximator)**.** *We define the approximator $D^*_{\mathcal{T}}$ over distributed belief pairs as follows:*

$$D^*_{\mathcal{T}}(\mathcal{B}) = ((\{I \mid (\mathcal{T}_A)^{\mathcal{B},I} \neq \mathbf{f}\})_{A\in\mathcal{A}}, (\{I \mid (\mathcal{T}_A)^{\mathcal{B},I} = \mathbf{t}\})_{A\in\mathcal{A}})$$

Definition 4.33 ($D^{st}_{\mathcal{T}}$ Operator)**.** *The stable operator $D^{st}_{\mathcal{T}}$ is the least fixpoint of the conservative bound given a fixed liberal bound, that is, $D^{st}_{\mathcal{T}}(\mathcal{Q}) = \mathtt{lfp}(D^*_{\mathcal{T}}(\cdot, \mathcal{Q})^c)$*

Definition 4.34 ($D^{st*}_{\mathcal{T}}$ Approximator)**.** *We define the approximator $D^{st*}_{\mathcal{T}}$ over a belief pair $\mathcal{B}$ as follows:*

$$D^{st*}_{\mathcal{T}}(\mathcal{B}) = (D^{st}_{\mathcal{T}}(\mathcal{B}^l), D^{st}_{\mathcal{T}}(\mathcal{B}^c))$$

As in the case for AEL, different fixpoints of these operators lead to different semantics to be defined.

Definition 4.35 (Supported Model)**.** *The dpws $\mathcal{Q}$ is a* supported model *of $\mathcal{T}$ iff $\mathcal{Q}$ is a fixpoint of $D_{\mathcal{T}}$.*

Definition 4.36 (Kripke-Kleene Model)**.** *The distributed belief pair $\mathcal{B}$ is the* Kripke-Kleene model *of $\mathcal{T}$ iif $\mathcal{B}$ is the least precise fixpoint of $D^*_{\mathcal{T}}$.*

Definition 4.37 (Stable Model)**.** *The dpws $\mathcal{Q}$ is a* stable model *of $\mathcal{T}$ iff $\mathcal{Q}$ is a fixpoint of $D^{st}_{\mathcal{T}}$.*

Definition 4.38 (Partial Stable Model)**.** *The distributed belief pair $\mathcal{B}$ is a* partial stable model *of $\mathcal{T}$ iff $\mathcal{B}$ is a fixpoint of $D^{st*}_{\mathcal{T}}$.*

Definition 4.39 (Well-Founded Model)**.** *The distributed belief pair $\mathcal{B}$ is the* well-founded model (*wfm*) *of $\mathcal{T}$ iff $\mathcal{B}$ is the least precise fixpoint of $D^{st*}_{\mathcal{T}}$.*

The following example shows how the different semantics for dAEL(ID) can be calculated. Again, this is a minimal example and for simplicity we do not include any inductive definition.

Example 4.6 (Candy)**.** *Suppose two principals, the mother and father of a child. A common scenario is one where the child fancies candy. The father says "You can have some candy if your mother says it is okay" while the mother says "You can have some candy if your father says it is okay". We model these statements in dAEL(ID) as:*

$$\mathcal{T}_D = \{M \textit{ says } c \Rightarrow c\} \qquad\qquad \mathcal{T}_M = \{D \textit{ says } c \Rightarrow c\}$$

If there is no further information available (i.e. which semantics to use) the child now can choose between the various semantics. We analyze all possible semantic to help the child choose. There exists four possible pws's for each agent:

(1) The lack of knowledge: $\bot_{\{D,M\}} = \{\{c\}, \{\}\}$

(2) The belief of c: $\mathcal{Q}^1_{\{D,M\}} = \{\{c\}\}$

(3) The disbelief of c: $\mathcal{Q}^2_{\{D,M\}} = \{\{\}\}$

(4) The empty pws or inconsistent belief: $\top_{\{D,M\}} = \emptyset$

Supported Model: *There are two supported models:* $(\bot_D, \bot_M)$ *and* $(\{\{c\}\}_D, \{\{c\}\}_M)$.

Kripke-Kleene Model: *The Kripke-Kleene model is:* $((\bot, \{\{c\}\})_D, (\bot, \{\{c\}\})_M)$.

Stable Model: *The stable model is:* $(\bot_D, \bot_M)$.

Partial Stable Model: *The partial stable model is:* $((\bot_D, \bot_M), (\bot_D, \bot_M))$.

Well-Founded Model: *The well-founded model is:* $((\bot_D, \bot_M), (\bot_D, \bot_M))$

4.4 Decision Procedure for dAEL(ID)

In this section, we motivate the minimization of the information flow between principals. We introduce the concepts required to build our decision procedure for the well-founded semantics of dAEL(ID). Finally, we provide a proof sketch for the correctness of the decision procedure.

4.4.1 IDP Inferences

The decision procedure defined in the next section is based on the IDP system.

We now define the two IDP inferences that we make use of. The first one, which is called `sat` in the IDP system, determines whether a given finite partial structure is a partial model of a given theory:

Definition 4.40. *Let S be a partial structure and $\mathcal{T}$ an* FO(ID) *theory. We say S is a* partial model *for $\mathcal{T}$ if and only if there exists a total structure $S' \geq_p S$ such that $S' \models \mathcal{T}$.*

The second IDP inference that we make use of, which is called `unsatstructure` in the IDP system, picks a minimal partial structure inconsistent with a given theory and less precise than a given finite partial structure:

Definition 4.41. *Let S be a partial structure and $\mathcal{T}$ be an* FO(ID) *theory. We define min_incons_set$(\mathcal{T}, S)$ to be the set of $\leq_p$-minimal partial structures $S' \leq_p S$ such that S' is not a partial model of $\mathcal{T}$.*

If the input structure S is not a partial model of the input structure $\mathcal{T}$, $min\_incons\_set(\mathcal{T}, S)$ is always non-empty, and `unsatstructure` picks an element from it and returns it. If S is a partial model of $\mathcal{T}$ `unsatstructure` throws an error.

4.4.2 Decision Procedure

In this section, we define a query-driven decision procedure for the well-founded semantics of dAEL(ID), which allows to determine access rights while minimizing the information flow between principals in order to enhance security and reduce privacy concerns. This decision procedure is implemented with the help of the IDP system. Given that IDP can only work with finite domains, the decision procedure also assumes the domain D to be finite.[a] For simplicity, we assume that for every principal there is a constant symbol referring to that principal, and that the t in every formula of the form $t\ says\ \varphi$ is such a constant symbol. This simplification could be removed, but would make the description of the decision procedure much more complicated.

The decision procedure is query-driven in the following sense: A query in the form of a dAEL(ID) formula φ is posed to a principal A. A determines whether her theory contains enough information in order to verify φ. It can happen that A cannot verify φ just on the basis of her theory, but can determine that if a certain other principal supports a certain formula, her theory implies the query. For example, A's theory may contain the formula $B\ says\ p \Rightarrow \varphi$. In this case, A can forward a remote sub-query to B concerning the status of p in B's theory. If B verifies the sub-query p and informs A about this, A can complete her verification of the original query φ.

[a] Given that propositional logic has the same expressive power as first-order logic over a finite domain, the decision procedure could in theory also be viewed as a decision procedure for the propositional fragment of dAEL(ID). But since first-order logic over a finite domain can model the same scenarios more concisely and more naturally than propositional logic, we stick to the first-order variant of dAEL(ID) with a finite-domain assumption.

4.4.3 Motivation for minimization of information flow

Consider the following distributed theory of the two principals A and B:

$$T_A = \left\{ \begin{array}{l} r \wedge B \; says \; s \Rightarrow p \\ r \end{array} \right\} \qquad T_B = \left\{ \begin{array}{l} s \\ \neg s \wedge A \; says \; p \Rightarrow p \end{array} \right\}$$

In both theories we have a guard, namely, r for theory T_A and $\neg s$ for theory T_B. The guards can be checked locally before performing a remote query to other theories. If A is queried about p, we can continue with the evaluation and query T_B about the truth value of s, since the guard r is true. If B is queried about p, on the other hand, we do not need to perform any remote query since it will always fail due to the guard being false in the theory.

If B nevertheless were to send the remote subquery p to A, this would be an unnecessary sub-query. Since B does not actually need to know whether A supports p, this would violate the need-to-know principle [SS94], which states that a principal should only be given those accesses and be provided with those non-public informations which the principal requires to carry out her responsibilities. Additionally, it is reasonable to assume that for privacy considerations, the principals do not want to disclose their full access control policies to other principals, but only the parts that are required to verify a given access request. So there are both security and privacy reasons for B not to send the remote subquery p to A.

In general, more complex behaviors rather than guards can occur in a distributed theory. The decision procedure we define minimizes the communication even when more complex reasoning is required to determine which sub-queries have a chance of leading to a verification of the primary query and which subqueries are certainly not useful. As discussed in Subsection 4.4.7, this ideal minimization of the communication is computationally very expensive, so in a practically applicable system, a trade-off between the security and privacy motivation for minizing communication on the one hand and computational cost on the other hand would need to be found. Nevertheless,

we consider our ideal minimization of communication an interesting proof of concept as a foundation for further research.

The decision procedure that determines whether a query α is true given a distributed theory $\mathcal{T}$ is composed of two distinct modules. The first module, the *Query Minimization Procedure*, looks at the theory of the agent to whom the query is directed, and determines minimal sets of remote calls to other theories that could verify the query. The second module, the *Communication Procedure*, takes care of communication between the principals, including the handelling of the loops that may occur.

4.4.4 Query Minimization Procedure

Translation Mechanism

In order to implement a query mechanism for dAEL(ID) in IDP we need to translate dAEL(ID) theories to FO(ID) theories. The only syntactic construct of dAEL(ID) that does not exist in FO(ID) is the *says*-modality. So when translating a dAEL(ID) theory T to an FO(ID) theory $\mathcal{T}$, we need to replace each *says*-atoms in T by some first-order formula. For this purpose, we extend the vocabulary Σ to an extended vocabulary Σ' by adding to it new propositional variables of the form $p^+_{A\_\texttt{says}\_\varphi}$, $p^-_{A\_\texttt{says}\_\varphi}$ and $w_{A\_\texttt{says}\_\varphi}$ for every modal statement $A\,says\,\varphi$ of dAEL(ID).

Before we formally define the translation mechanism, let us first motivate why we have the three different propositional variables $p^+_{A\_\texttt{says}\_\varphi}$, $p^-_{A\_\texttt{says}\_\varphi}$ and $w_{A\_\texttt{says}\_\varphi}$ for translating different occurrences of the same *says*-atom $A\,says\,\varphi$. First, note that the well-founded semantics of dAEL(ID) evaluates *says*-atoms in a three-valued way. The propositional variables $p^+_{A\_\texttt{says}\_\varphi}$ and $p^-_{A\_\texttt{says}\_\varphi}$ are used to model the three-valued valuation of $A\,says\,\varphi$ in the two-valued logic FO(ID): On the precision order $<_p$ on the three truth values $\mathbf{t}$ (*true*), $\mathbf{f}$ (*false*) and $\mathbf{u}$ (*undefined*) induced by $\mathbf{u} <_p \mathbf{t}$ and $\mathbf{u} <_p \mathbf{f}$, the propositional variable $p^+_{A\_\texttt{says}\_\varphi}$ represents the upper bound for the truth value of $A\,says\,\varphi$ and $p^-_{A\_\texttt{says}\_\varphi}$ the lower bound. For this reason, we replace

every positive occurrence of $A\ says\ \varphi$ by $p^{+}_{A\_\texttt{says}\_\varphi}$, and every negative occurrence by $p^{-}_{A\_\texttt{says}\_\varphi}$. Given that occurrences of a formula in an inductive definition cannot be meaningfully termed only positive or only negative, we first replace occurrences of $A\ says\ \varphi$ in an inductive definition by $w_{A\_\texttt{says}\_\varphi}$ and add two implications to the theories that express the equivalence between $w_{A\_\texttt{says}\_\psi}$ and $A\ says\ \varphi$.

The translation function t only performs this first step of the translation mechanism:

Definition 4.42. *Let T be a dAEL(ID) theory. We define $t(T)$ to be a dAEL(ID) theory equivalent to T, constructed as follows:*

For every modal atom $A\ says\ \varphi$ occurring in the body of an inductive definition in theory T,

- *replace $A\ says\ \varphi$ by the propositional variable $w_{A\_\texttt{says}\_\varphi}$*
- *add to $t(T)$ the two formulae $w_{A\_\texttt{says}\_\varphi} \Rightarrow A\ says\ \varphi$ and $A\ says\ \varphi \Rightarrow w_{A\_\texttt{says}\_\varphi}$.*

We next introduce the notion of polarity necessary to further translate dAEL(ID) theories into FO(ID) theories.

Definition 4.43. *Let φ be a dAEL(ID) formula. The* polarity *of an occurrence of a subformula of φ is defined recursively as follows:*

- *The occurrence of φ in φ is a* positive *occurrence.*
- *Given a positive (resp.* negative*) occurrence of the the subformula $\neg\psi$ of φ, the occurence of ψ in this occurrence of $\neg\psi$ is* negative *(resp.* positive*) in φ.*
- *Given a* positive *(resp.* negative*) occurrence of the subformula $\psi \wedge \chi$ of φ, the occurrences of ψ and χ in this occurrence of $\psi \wedge \chi$ are both* positive *(resp.* negative*) in φ.*

Definition 4.44. *Let T be a dAEL(ID) theory, let $\varphi \in T$. We call a positive (resp. negative) occurrence of a subformula ψ of φ a* positive *(resp.* negative*)* occurrence *of ψ in T.*

Now we can define the translation function τ from dAEL(ID) theories to FO(ID) theories:

Definition 4.45. *Let T be a dAEL(ID) theory. $\tau(T)$ is constructed from $t(T)$ by performing the following replacements for every says-atom A says φ occurring in $t(T)$ that is not the subformula of another says-atom:*

- *Replace every positive occurrence of A says φ in T by $p^+_{A\_\texttt{says}\_\varphi}$.*
- *Replace every negative occurrence of A says φ in T by $p^-_{A\_\texttt{says}\_\varphi}$.*

We will illustrate the translation procedure with a simple example, which we will use as a running example to be extended throughout the section.

Example 4.7. *Let $\mathcal{A} = \{A, B, C\}$, and let the distributed theory $\mathcal{T}$ consist of the folowing three dAEL(ID) theories:*

$$T_A = \left\{ \begin{array}{l} \{\ p \leftarrow B\ \ says\ \ p, \\ p \leftarrow r\ \} \\ p \wedge s \wedge B\ \ says\ \ z \Rightarrow z \\ r \vee \neg r \Rightarrow s \\ B\ \ says\ \ r \vee \neg(B\ \ says\ \ r) \Rightarrow z \end{array} \right\} \quad T_B = \left\{ \begin{array}{l} p \\ C\ \ says\ \ z \Rightarrow z \\ C\ \ says\ \ r \Rightarrow r \end{array} \right\}$$

$$T_C = \left\{ \begin{array}{l} \neg(B\ \ says\ \ z) \Rightarrow z \\ B\ \ says\ \ r \Rightarrow r \end{array} \right\}$$

We translate these theories as follows:

$$\tau(T_A) = \left\{ \begin{array}{l} \{\ p \leftarrow w_{B\_says\_p}, \\ p \leftarrow r\ \} \\ w_{B\_says\_p} \Rightarrow p^+_{B\_says\_p} \\ p^-_{B\_says\_p} \Rightarrow w_{B\_says\_p} \\ p \wedge s \wedge p^-_{B\_says\_z} \Rightarrow z \\ r \vee \neg r \Rightarrow s \\ p^-_{B\_says\_r} \vee \neg p^+_{B\_says\_r} \Rightarrow z \end{array} \right\} \quad \tau(T_B) = \left\{ \begin{array}{l} p \\ p^-_{C\_says\_z} \Rightarrow z \\ p^-_{C\_says\_r} \Rightarrow r \end{array} \right\}$$

$$\tau(T_C) = \left\{ \begin{array}{l} \neg p^+_{B\_says\_z} \Rightarrow z \\ p^-_{B\_says\_r} \Rightarrow r \end{array} \right\}$$

Query Minimization Procedure

The query minimization procedure works as follows: given a theory T and a query α, the procedure returns a set $\mathbb{L}$ of sets of modal atoms. The intended meaning of $\mathbb{L}$ is as follows: When all modal atoms in a set $L \in \mathbb{L}$ can be determined to be true, the query α succeeds, and $\mathbb{L}$ is the set of all sets L with this property. This means that if $\mathbb{L} = \emptyset$, the query necessarily fails, whereas if $\mathbb{L}$ contains $\emptyset$, the query necessarily succeeds.

A partial structure S over the extended vocabulary Σ' contains information about the truth values of the propositional variables of the form $p^-_{A\_\mathtt{says}\_\varphi}$ and $p^+_{A\_\mathtt{says}\_\varphi}$. Taking into account that $p^-_{A\_\mathtt{says}\_\varphi}$ and $p^+_{A\_\mathtt{says}\_\varphi}$ are used to represent the three-valued valuation of $A\,says\,\varphi$, this information can also be represented by a set of *says*-literals, which we denote L^S:

Definition 4.46. *For a partial structure $S = (D, \mathcal{I})$, we define L^S to be*

$$\{A\,says\,\varphi \mid (p^-_{A\_\mathtt{says}\_\varphi})^{\mathcal{I}} = \mathbf{t}\} \cup \{\neg A\,says\,\varphi \mid (p^+_{A\_\mathtt{says}\_\varphi})^{\mathcal{I}} = \mathbf{f}\}$$

We say that a *says*-atom $A\,says\,\varphi$ *occurs directly* in a dAEL(ID) theory, if some occurrence of $A\,says\,\varphi$ in $\mathcal{T}$ is not a subformula of another *says*-atom. In the Query Minimization Procedure, we need to take into account all possible three-valued valuations of the *says*-atoms directly occurring in the input dAEL(ID) theory T. Such a valuation can be represented by a partial structure that contains information only about propositional variables of the form $p^+_{A\_\mathtt{says}\_\varphi}$ and $p^-_{A\_\mathtt{says}\_\varphi}$, and for which this information is coherent in the sense that the truth values assigned to $p^+_{A\_\mathtt{says}\_\varphi}$ and $p^-_{A\_\mathtt{says}\_\varphi}$ are compatible. This is made formally precise in the following definition of the set $\mathbb{S}_T$ that contains all structures that represent three-valued valuations of *says*-atoms directly occurring in T:

Definition 4.47. *Let T be a dAEL(ID) theory. We define $\mathbb{S}_T$ to be the set containing every partial structure $S = (D, \mathcal{I})$ over vocabulary Σ' satisfying the following properties:*

- $P^{\mathcal{I}^S} = \mathbf{u}$ *for every symbol in Σ' that is not of the form $p^+_{A\_says\_\varphi}$ or $p^-_{A\_says\_\varphi}$ for some says-atom $A\,says\,\varphi$ occurring in $\tau(T)$.*
- *For every says-atom $A\,says\,\varphi$, $(p^+_{A\_says\_\varphi})^{\mathcal{I}} \neq \mathbf{t}$.*
- *For every says-atom $A\,says\,\varphi$, $(p^-_{A\_says\_\varphi})^{\mathcal{I}} \neq \mathbf{f}$.*
- *For no says-atom $A\,says\,\varphi$, $(p^+_{A\_says\_\varphi})^{\mathcal{I}} = \mathbf{f}$ and $(p^-_{A\_says\_\varphi})^{\mathcal{I}} = \mathbf{t}$.*

We are now ready to define the Query Minimization Procedure. Its pseudo-code is as follows (Algorithm 1).

Algorithm 4 Query Minimization Procedure

Require: theory T, dAEL(ID) query α
Ensure: set $\mathbb{L}$ of sets of modal atoms
1: $\mathbb{L} := \emptyset$
2: $\mathcal{T} := \tau(T)$
3: **for each** $S \in \mathbb{S}_T$ **do**
4: **if** S is not a partial model of $\mathcal{T} \cup \{\neg\alpha\}$ **then**
5: pick a partial structure S_{min} from $min\_incons\_set(\mathcal{T} \cup \{\neg\alpha\}, S)$
6: $\mathbb{L} := \mathbb{L} \cup \{L^{S_{min}}\}$
7: **return** $\mathbb{L}$

The algorithm is to be read as follows. A query α asked to theory T is given as input. First (line 2) we translate T to the FO(ID) theory $\mathcal{T} = \tau(T)$. Next we iterate over the structures $S \in \mathbb{S}_T$ (lines 3-6). Line 4 ensures that we limit ourselves to structures $S \in \mathbb{S}_T$ that are not a partial models of $\mathcal{T} \cup \{\neg\alpha\}$; note that the information in such a structure S together with the information in $\mathcal{T}$ entails the query α. Furthermore, note that for such a structure S, $min\_incons\_set(\mathcal{T} \cup \{\neg\alpha\}, S)$ is non-empty. So next (line 5), we pick a structure S_{min} from $min\_incons\_set(\mathcal{T} \cup \{\neg\alpha\}, S)$; by definition S_{min} is a minimal structure such that $S_{min} \leq_p S$ and S_{min} is not a partial

model of $\mathcal{T} \cup \{\neg\alpha\}$; this means that S_{min} contains a minimal amount of information from S that together with the information in $\mathcal{T}$ ensures the query α to be true. So the set $L^{S_{min}}$, which represents the same information as a set of *says*-literals, is a minimal set of *says*-literals that together with the information in T ensure the query α to be true.[b] Line 6 adds $L^{S_{min}}$ to the set of sets of *says*-literals that we output at the end (line 7), after the iteration over the elements of $\mathbb{S}_T$ is completed.

We continue Example 4.7 to illustrate the query minimization procedure.

Example 4.8. *We apply the Query Minimization Procedure to the theory T_A and the query z. First we translate the theory T_A to $\mathcal{T} = \tau(T_A)$, which we have shown in Example 4.7. Then we iterate over the structures $S \in \mathbb{S}_{T_A}$.*

Let, for example, S be the element of $\mathbb{S}_{T_A}$ that makes $p^-_{B\_says\_p}$ and $p^-_{B\_says\_r}$ true and everything else undefined. Then S is a not partial model of $\mathcal{T} \cup \{\neg z\}$, because $p^-_{B\_says\_p}$ is inconsistent with $p^-_{B\_says\_r} \vee \neg p^+_{B\_says\_r} \Rightarrow z$ and z. Now $min\_incons\_set(\mathcal{T} \cup \{\neg z\}, S)$ is the set consisting only of the structure S' that makes $p^-_{B\_says\_p}$ true and everyting else undefined. So in line 5, we necessarily pick S_{min} to be this structure S'. In line 6 we calculate $L^{S'}$ to be $\{B\ says\ r\}$ and add $\{B\ says\ r\}$ to $\mathbb{L}$.

When we iterate over all structures $S \in \mathbb{S}_{T_A}$, the value of $\mathbb{L}$ finally becomes

$$\{\{B\ says\ r\}, \{B\ says\ p, B\ says\ z\}, \{\neg B\ says\ r\}\}.$$

4.4.5 Communication and loop handling

In this subsection we describe the Communication Procedure, which also takes care of the loop-handling. The Communication Procedure calls the Query Minimization Procedure and thereby constitutes our decision procedure for dAEL(ID).

When a query is asked to a principal, the Query Minimization Procedure determines minimal sets of *says*-literals that need to be satisfied in order to verify

[b] Lemma 4.1 makes this claim more precise.

the query. The Communication Procedure then produces remote sub-queries to other principals that can determine the status of the *says*-literals.

The Communication Procedure works by dynamically producing a *query graph* and attaching three-valued truth values to the query vertices in it:

Definition 4.48. *A* query graph *is a labelled directed graph with two kinds of vertices and two kinds of edges:*

- *The first kind of vertices are the* query vertices. *Each query vertex is labelled by a directed query of the form $\langle k : \varphi \rangle$, where k is the principal whose theory is being queried and φ is the formula representing the query. Additionally, a query vertex is potentially labelled by a truth value in $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, which represents the currently active valuation of the query at any moment during the execution of the decision procedure.*

- *The second kind of vertices are the says*-literal set vertices. *Each says-literal set vertex is labelled by a set of says-literals, i.e. formulas of the form k says φ or $\neg k$ says φ.*

- *The first kind of edges are unlabelled edges going from a query vertex to a says-literal set vertex. The intended meaning of such an unlabelled edge from $\langle k : \alpha \rangle$ to the says-literal set L is that one way of making α true in k's theory is to make all says-literals in L true.*

- *The second kind of edged are edges labelled by* $\mathbf{t}$ *or* $\mathbf{f}$*, going from a says-literal set vertex to a query vertex. The intended meaning of such an edge labelled by* $\mathbf{t}$ *or* $\mathbf{f}$ *and going from the says-literal set L to the query $\langle k : \alpha \rangle$ is that L contains the literal k says α or the literal $\neg k$ says α respectively.*

The query graphs are actually always trees, with the query vertex corresponding to the original query as their root.

The Communication Procedure starts with a query graph consisting just of the query vertex $\langle A : \alpha \rangle$, where A is the principal to whom the primary query α is asked.

Next the Communication Procedure calls the Query Minimization Procedure to add sub-queries to the query graph and attach truth values to them. This procedure is iteratively continued until a truth-value has been attached to the root vertex $\langle A : \alpha \rangle$.

The Communication Procedure is defined via an initialization procedure defined under Algorithm 5, which calls the main recursive procedure defined under Algorithm 6.

Algorithm 5 Communication Procedure Initialization

Require: distributed theory $\mathcal{T}$, principal A, dAEL(ID) formula α
Ensure: truth-value $V \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$
1: $G :=$ the labelled graph consisting only of a single vertex v labelled $\langle A : \alpha \rangle$ and no edges
2: $G :=$ Communication_Procedure($\mathcal{T}$,G,v)
3: $V :=$ the label on the query vertex $\langle A : \alpha \rangle$ in G
4: **return** V

Informally, the Communication Procedure can be explained as follows: The Query Minimization Procedure is called for T_A and α. It returns a set of sets of *says*-literals. For each such *says*-literal set, we add a *says*-literal set vertex connected to the root query vertex $\langle A : \alpha \rangle$ (lines 6-7). For each *says*-literal in this set, we add a query vertex and an edge from the set vertex to this query vertex labelled by **t** or **f** depending on the sign of the *says*-literal (8-15). We then apply Query Minimization Procedure and the rest of the procedure just explained to each new query vertex (line 22). At the same time, we label query vertices with truth values as follows: When all query vertices emerging from a *says*-literal set vertex are labelled with the same truth value as the edge through which they are connected to the *says*-literal set vertex, the query that produced that *says*-literal set vertex is labelled **t** (lines 23-24). There is a dual procedure for labelling query vertices with **f** (lines 25-26). When a loop is detected, the query vertex causing the loop (by having the same label as a query vertex that is an ancestor of it) is labelled either with **f** or **u**, depending on whether the loop is over a negation (i.e. there is an **f**-labelled edge in the path connecting the two vertices

with the same label) or not (lines 16-20). **u**-labels can also propagate towards the root of the graph (line 28).

Algorithm 6 Communication Procedure

Require: distributed theory $\mathcal{T}$, query graph G, query vertex v of G,
Ensure: updated query graph G

```
1: k := the principal mentioned in the label of v
2: φ := the formula mentioned in the label of v
3: 𝕃 := Query_Minimization_Procedure(T_k,φ)
4: while the input query vertex v does not have a truth-value attached to it do
5:   for L ∈ 𝕃 do
6:     add a new says-literal set vertex L to G
7:     add to G a new edge from vertex v to vertex L
8:     for l ∈ L do
9:       k' := the principal such that l is of the form k' says ψ or ¬k' says ψ
10:      ψ := the formula such that l is of the form k' says ψ or ¬k' says ψ
11:      add a query vertex v' labelled by ⟨k' : ψ⟩ to G
12:      if l is k' says ψ then
13:        add to G a new edge labelled t from vertex L to vertex ⟨k' : ψ⟩
14:      if l is ¬k' says ψ then
15:        add to G a new edge labelled f from vertex L to vertex ⟨k' : ψ⟩
16:      if a query vertex v'' that is an ancestor of v' is also labelled ⟨k' : ψ⟩ then
17:        if all labelled edges between v'' and v' are labelled by t then
18:          add f-label to v'
19:        else
20:          add u-label to v'
21:      else
22:        Communication_Procedure(T,G,v')
23:    if every query vertex v' such that there is an edge from L to v' is labelled
       with the same truth value as this edge then
24:      label v with t
25:  if for every says literal set vertex L such that there is an edge from v to L,
       there is a query vertex v' such that there is an edge from L to v' labelled with
       the opposite truth value as v' then
26:    label v with f
27:  else
28:    label v with u
29: return G
```

We continue Example 4.7 to illustrate the Communication Procedure.

Example 4.9. *Given the distributed theory* $T = \{T_A, T_B, T_C\}$*, we query the principal* A *about the truth value of* z*. We show the final graph in Figure 4.1 and now explain its construction. We start by calling the Communication Initialization Procedure; this generates a graph* G *with the vertex* $v = \langle A : z\rangle$ *(with no associated truth value label). Then we call the Communication Procedure with arguments* $\mathcal{T}$*,* G *and* v*. We call the Query Minimization Procedure returning the set* $\mathbb{L} = \{\{B \text{ says } p, B \text{ says } z\}, \{B \text{ says } r\}, \{\neg B \text{ says } r\}\}$ *as shown in Example 4.8. Since the input vertex* v *has no truth-value associated to it, we next iterate over the sets* $L \in \mathbb{L}$*. The says-literal set vertex* $\{B \text{ says } p, B \text{ says } z\}$ *is added to* G *with its corresponding edge. Now we consider each says-literal in the vertex.*

(i) For literal $B \text{ says } p$ *generate a new query vertex* $v' = \langle B : p\rangle$ *with an edge labelled* $\mathbf{t}$ *(since the literal is not negated) and recursively call the Communication Procedure with the updated graph as argument and vertex* v'*.* v' *has no truth-value associated, the Query Minimization Procedure returns the set* $\mathbb{L}' = \{\{\}\}$*; so after adding the says-literal set vertex* $\{\}$ *connected to* v'*, the truth-value* $\mathbf{t}$ *is assigned to* v' *by lines 21-22 of the Communication Procedure (this corresponds to the intuitive idea that* $\mathbb{L}' = \{\{\}\}$ *means that* p *is true in* T_B*).*

(ii) For literal $B \text{ says } z$ *generate a new query vertex* $v' = \langle B : z\rangle$ *with an edge labelled* $\mathbf{t}$ *and recursively call the Communication Procedure with the updated graph as argument and vertex* v'*. The Query Minimization Procedure is called returning the set* $\mathbb{L} = \{\{C \text{ says } z\}\}$*; for this literal we generate a new query vertex* $v'' = \langle C : z\rangle$ *with an edge labelled* $\mathbf{t}$*. In turn, the Query Minimization Procedure is called returning the set* $\mathbb{L}'' = \{\{\neg B \text{ says } z\}\}$*; for this literal we generate a new query vertex* $v''' = \langle B : z\rangle$ *with an edge labelled* $\mathbf{f}$*. At this point we detect a loop, as the query vertex* v' *that is an ancestor of* v''' *is also labelled by* $\langle B : z\rangle$*. Since the loop contains an edge with label* $\mathbf{f}$*, the truth-value assignment for* v''' *is* $\mathbf{u}$*. This truth-value* $\mathbf{u}$ *is propagated up to label the query vertices* v'' *and* v'*, since* $\mathbf{u}$ *does not match with either* $\mathbf{t}$ *nor* $\mathbf{f}$*.*

Finally, the truth-values for (i) matches the labeled edge, but not for the case of (ii). Thus we cannot yet label the root vertex with $\mathbf{t}$*, and continue with the next says-literal* $\{B \, says \, r\} \in \mathbb{L}$*.*

For vertex $\{B \, says \, r\}$*, we repeat the procedure as described above until we (again) detect a loop. This loop does not contain edges with label* $\mathbf{f}$*, so the truth-value assignment for the vertex at which the loop is detected is* $\mathbf{f}$*. Again this truth-value is propagated to label the two query vertices above this vertex, as the labelled edges are labelled by* $\mathbf{t}$*. Since the truth-value* $\mathbf{f}$ *assigned to the query vertex* $\langle B : r \rangle$ *does not match the truth value of the labelled edge above it, the root vertex can still not be labelled with* $\mathbf{t}$*.*

The subgraph produced below the final says-literal set vertex $\{\neg B \, says \, r\}$ *is the same as below says-literal set vertex* $\{B \, says \, r\}$*, only that the labelled edge directly below this says-literal set vertex is now labelled* $\mathbf{f}$ *instead of* $\mathbf{t}$*. So this time the label on the query vertex* $\langle B : r \rangle$ *matches the label on the labelled edge, to that the root vertex is labelled* $\mathbf{t}$*. This ends the main while loop and therefore the Communication Procedure. Finally, the Communication Procedure Initialization returns the output* $\mathbf{t}$*.*

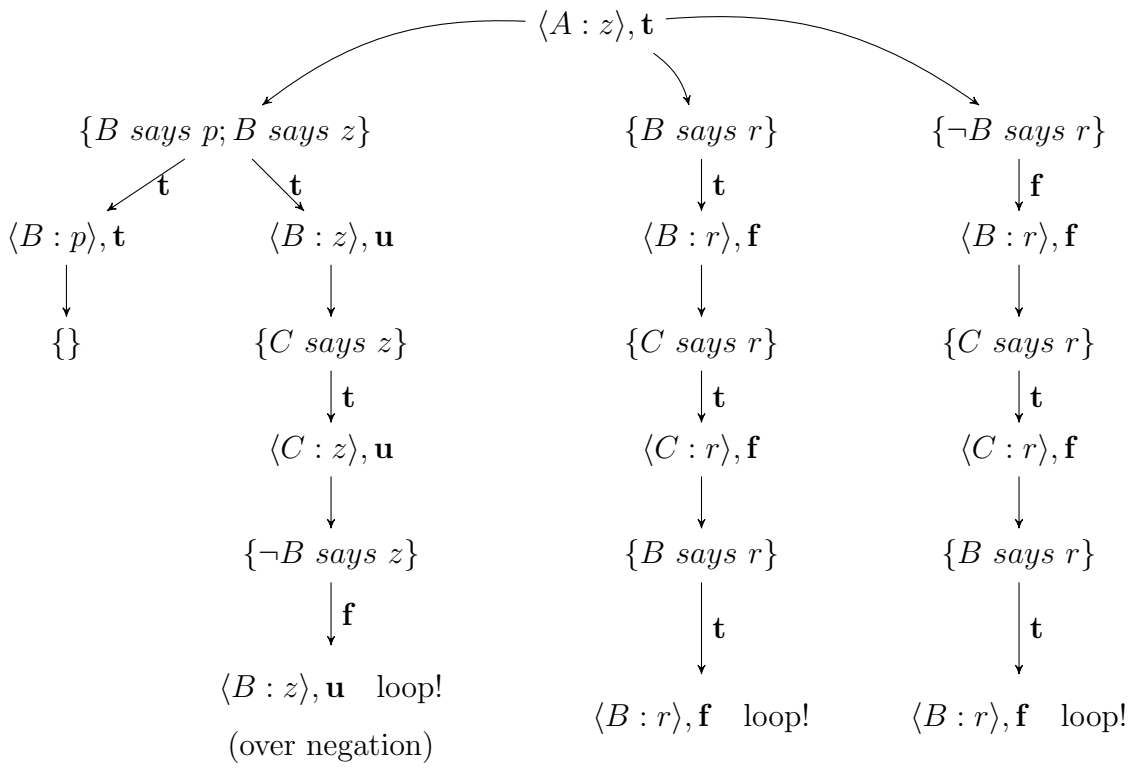


Figure 4.1: Query Graph

4.4.6 Correctness of decision procedure

In this section, we provide definitions and lemmas necessary to prove the correctness of the decision procedure. The proofs sketched in this section were provided by my collaborator Marcos Cramer.

First we need to establish that the Query Minimization Procedure (Algorithm 4) really does what it is supposed to do, namely to return the set of all minimal sets of *says*-literals that make the query true. In order to formalize the notion of a set of *says*-literals that makes the query true, we first need the following definitions of three-valued valuations of dAEL(ID) formulas, inductive definitions and theories with respect to a set of *says*-literals and a structure. These definitions are analogous to the definitions 4.29, 4.30 and 4.31:

Definition 4.49. *We inductively define a three-valued valuation of dAEL(ID) formulas with respect to a set L of says-literals and a structure I as follows:*

$$(P(\bar{t}))^{L,I} = \begin{cases} \mathbf{t} & \text{if } t^I \in P^I \\ \mathbf{f} & \text{if } t^I \notin P^I \end{cases}$$

$$(\neg\varphi)^{L,I} = (\varphi^{L,I})^{-1}$$

$$(\varphi \wedge \psi)^{L,I} = glb_{\leq_t}(\varphi^{L,I}, \psi^{L,I})$$

$$(\forall x\ \varphi)^{L,I} = glb_{\leq_t}\{\varphi[x/d]^{L,I} \mid x \in D\}$$

$$(A\ says\ \varphi)^{L,I} = \begin{cases} \mathbf{t} & \text{if } A \in \mathcal{A} \text{ and} \\ & \quad (A\ says\ \varphi) \in L \\ \mathbf{f} & \text{if } A \notin \mathcal{A} \text{ or} \\ & \quad (\neg A\ says\ \varphi) \in L \\ \mathbf{u} & \text{otherwise} \end{cases}$$

Definition 4.50. *We define a three-valued valuation of dAEL(ID) inductive definitions with respect to a set L of says-literals and a structure I as follows:*

$$\Delta^{L,I} = \begin{cases} \mathbf{t} & \text{if } I = wfm_{\Delta^L}(I|_{Par(\Delta)}) \\ \mathbf{f} & \text{if } I \not\geq_p wfm_{\Delta^L}(I|_{Par(\Delta)}) \\ \mathbf{u} & \text{otherwise} \end{cases}$$

where Δ^L is the definition Δ with every formula $A\ says\ \varphi$ replaced by $\mathbf{t}$, $\mathbf{f}$ or $\mathbf{u}$, according to whether $(A\ says\ \varphi) \in L$, $(\neg A\ says\ \varphi) \in L$ or neither of them is in L.

Definition 4.51. *We define a three-valued valuation of dAEL(ID) theories with respect to a distributed belief pair L and a structure I as follows:*

$$T^{L,I} := glb_{\leq_t}(\{\varphi^{L,I} | \varphi \in T\} \cup \{\Delta^{L,I} | \Delta \in T\})$$

Now we can define what it means for a set L of *says*-literals to make true a formula:

Definition 4.52. *Let L be a set of says-literals, let φ be a dAEL(ID) formula and let $\mathcal{T}$ be a dAEL(ID) theory. Then we say that L* makes *φ* true relative to *T iff for every structure I such that $T^{L,I} \neq \mathbf{f}$ we have $\varphi^{L,I} = \mathbf{t}$.*

The following lemma states that the Query Minimization Procedure (Algorithm 4) really does what it is supposed to do:

Lemma 4.1. *Let T be a dAEL(ID) theory and let α be a dAEL(ID) formula. The set $\mathbb{L}$ returned by* Query_Minimization_Procedure*(T, α) is*

$$\{L \mid L \text{ is minimal (under set inclusion) among the sets } L' \text{ of says-literals that make } \alpha \text{ true with respect to } T\}$$

Proof. For proving this lemma, we need to show that $L \in \mathbb{L}$ iff L is a minimal set that makes α true with respect to T.

First suppose $L \in \mathbb{L}$. Then by the definition of Algorithm 4, α and min_incons, L is a minimal set such that $\mathcal{T} \cup \{p^-_{A\_says\_\varphi} \mid (A\ says\ \varphi) \in L\} \cup \{\neg p^+_{A\_says\_\varphi} \mid (\neg A\ says\ \varphi) \in L\} \cup \{\alpha\} \models \bot$. Hence L is a minimal set such that $\mathcal{T} \cup \{p^-_{A\_says\_\varphi} \mid (A\ says\ \varphi) \in L\} \cup \{\neg p^+_{A\_says\_\varphi} \mid (\neg A\ says\ \varphi) \in L\} \models \alpha$. So by Definitions 4.49, 4.51 and 4.52 above, L is a minimal set that makes α true with respect to T, as required.

Conversely, suppose L is a minimal set that makes α true with respect to T. Define S to be the partial structure that makes α true, makes $p^-_{A\_says\_\varphi}$ true whenever $(A\ says\ \varphi) \in L$, that makes $p^+_{A\_says\_\varphi}$ false whenever $(\neg A\ says\ \varphi) \in L$, and that is undefined on all other symbols. Then applying lines 4-6 of Algorithm 4 to this structure S leads to L being added to $\mathbb{L}$, as required.

□

The well-founded model of $\mathcal{T}$ is the $\leq_p$-least fixpoint of $D^{st}_{\mathcal{T}}$. When the domain is finite, as we are assuming when applying the decision procedure, there is a natural number n such that $wfm(\mathcal{T}) = (D^{st}_{\mathcal{T}})^n(\bot, \top)$. In other words, the well-founded model

can be computed by a finite number of application of $D_{\mathcal{T}}^{st}$ to $(\bot, \top)$, until a fixpoint is reached.

The steps in the decision procedure defined in Section 4.4.5 do not directly correspond to the steps in the computation of the well-founded model by a finite number of application of $D_{\mathcal{T}}^{st}$ to $(\bot, \top)$. In order to prove that the two computations nevertheless always yield the same result, we first define a decision procedure that resembles the decision procedure defined in Section 4.4.5, but whose steps correspond more directly to the iterative application of $D_{\mathcal{T}}^{st}$ to $(\bot, \top)$. We call this auxiliary decision procedure the *$D_{\mathcal{T}}^{st}$-based decision procedure*. So we prove the correctness of the decision procedure in Section 4.4.5 by proving two things:

- The decision procedure defined in Section 4.4.5 is equivalent to the $D_{\mathcal{T}}^{st}$-based decision procedure.
- When A's theory $\mathcal{T}_A$ is queried about α, the $D_{\mathcal{T}}^{st}$-based decision procedure returns yes iff $(A \; says \; \alpha)^{wfm(T)} = \mathbf{t}$.

At this point the reader may wonder why we don't directly use the $D_{\mathcal{T}}^{st}$-based decision procedure as our decision procedure for the well-founded semantics. There are two reasons: The decision procedure defined in Section 4.4.5 is more efficient, and it is better at minimizing communication between the different agents' theories, which is a relevant feature for the access control application we have in mind.

In the definition of the $D_{\mathcal{T}}^{st}$-based decision procedure, we use a *query graph* as defined in section 4.4.5. Unlike the query graph produced in the decision procedure from Section 4.4.5, the labelling of the query vertices in the query graph produced by the $D_{\mathcal{T}}^{st}$-based decision procedure are unique, i.e. there are no two distinct query vertex with the same labelling $k : \varphi$.

There is a direct correspondence between distributed belief pairs and certain truth-value labelling of the query vertices in a query graph:

Definition 4.53. *Let G be a query graph. Let $\mathcal{B}$ be a distributed belief pair. We say that the truth-value labelling of the query vertices of G* corresponds to *$\mathcal{B}$ iff for each query vertex $k : \varphi$ in G, the truth-value with which this vertex is labelled is $(k \text{ says } \varphi)^{\mathcal{B}}$.*

Note that there are truth-labellings of the query vertices that do not correspond to any distributed belief pair. We call a truth-labelling of the query vertices *good* iff it corresponds to some distributed belief pair.

The $D^{st}_{\mathcal{T}}$-based decision procedure works by first producing a query graph and then iteratively modifying the truth-value labelling of the query vertices. We need to ensure that after each iteration of this iterative modification, the truth-value labelling of the query vertices is good. However, there are intermediate steps within each iteration which lead to a bad labelling of the query vertices. In order to get back to a good labelling, we apply the changes defined by Algorithm 7.

Algorithm 7 Make labelling of query vertices good

Require: query graph G
Ensure: modified query graph G
1: **while** there is a **u**-labelled query vertex $k : \varphi$ in G such that replacing *says*-atoms in φ corresponding to **t**- or **f**-labelled query vertices by **t** and **f** respectively makes φ a tautology **do**
2: change the **u**-label in each such query vertex in G by **t**
3: **while** there is a **f**-labelled query vertex $k : \varphi$ in G such that replacing *says*-atoms in φ corresponding to **u**-, **t**- or **f**-labelled query vertices by **t** or **f**, **t** and **f** respectively makes φ a tautology **do**
4: change the **f**-label in each such query vertex in G by **u**
5: **return** G

In order to define the $D^{st}_{\mathcal{T}}$-based decision procedure, we furthermore need the following two definitions:

Definition 4.54. *In a query graph, a says-literal set vertex L is defined to be* satisfied *if for every* **t***-labelled edge from L to a query vertex, the query vertex is labelled by* **t***, and for every* **f***-labelled edge from L to a query vertex, the query vertex is labelled* **f***.*

Definition 4.55. *In a query graph, a says-literal set vertex L is defined to be* potentially satisfied *if for every* **t***-labelled edge from L to a query vertex, the query*

vertex is labelled by **t** *or* **u**, *and for every* **f**-*labelled edge from* L *to a query vertex, the query vertex is labelled* **f** *or* **u**.

The definition of the $D^{st}_{\mathcal{T}}$-based decision procedure is given by the pseudo-code under Algorithm 8. Here is an informal explanation of this definition:

- First (lines 1-20) produce a query graph G without truth-value label on query vertices by iteratively applying the query minimization procedure from Section 4.4.5 to the main query A_α, to its subqueries, their subqueries etc., adding vertices and edges to the graph in line with the informal meaning of the vertices and edges given in Definition 4.48 above.

- Each query vertex in G is labelled **u** (line 21).

- Next (lines 22-36) we iteratively turn some of the **u**-labels on queries into **t**-labels and **f**-labels. At each step of this iteration, we separately calculate which **u**-labels to change to **t** and which **u**-labels to turn to **f** as follows:

    1. In order to calculate which **u**-labels to change to **t**, we recursively do the following (lines 25-27): When there is an edge from a query vertex labelled **u** to a satisfied *says*-literal set, change the label of the query vertex to **t**. Then make the labelling of the query vertices good. (All the queries whose labels change from **u** to **t** in this recursive procedure are changed to **t** in G in line 33.)

    2. In order to calculate which **u**-labels to change to **f**, we first change the labelling of all **u**-labelled queries to **f** (line 29) and then recursively do the following (lines 30-32): When there is an edge from a query vertex labelled **f** to a potentially satisfied *says*-literal set, change the label of the query vertex to **u**. Then make the labelling of the query vertices good. (All the queries whose labels stay **f** in this recursive procedure are changed to **f** in G in line 34.)

- Finally, we return the truth-value by which the query vertex of the main query $A : \alpha$ is labelled in G (lines 37-38).

Lemma 4.2. *Let $\mathcal{T}$ be a dAEL(ID) theory, A be a principal and α be a dAEL(ID) formula. The truth value returned by* Communication_Procedure_Initialization($\mathcal{T}$,A,α) *is equivalent to the truth value returned by* $D^{st}_{\mathcal{T}}$-based_decision_procedure($\mathcal{T}$,A,α)

Proof Sketch. The only fundamental difference between these two decision procedures is the loop-handling. Step 2) of the $D^{st}_{\mathcal{T}}$-based decision procedure takes care of making queries looping over **t**-labelled edges false. Queries looping over **f**-labelled edges will always be left undecided by the $D^{st}_{\mathcal{T}}$-based decision procedure, which corresponds to making them undecided in the decision procedure defined in section 4.4.5. □

We now establish that the $D^{st}_{\mathcal{T}}$-based decision procedure always gives the same result as the well-founded semantics. Note that the labelling corresponding to the distributed belief pair $(\bot, \top)$ is the labelling in which all query vertices are labelled by **u**. Keeping in mind that the well-founded model can be computed by a finite number of application of $D^{st}_{\mathcal{T}}$ to $(\bot, \top)$, it is now easy to see that the following lemma is sufficient to establish that the $D^{st}_{\mathcal{T}}$-based decision procedure always gives the same result as the well-founded semantics:

Lemma 4.3. *Let $\mathcal{T}$ be a distributed theory, A be a principal and α be a dAEL(ID) formula. Let G be the query graph produced by lines 1-20 of Algorithm 8 applied to $\mathcal{T}$, A and α. Let $\mathcal{B}$ be a distributed belief pair. Labelling the query vertices in G according to $\mathcal{B}$ and then applying lines 24 to 34 of Algorithm 8 to G yields a labelling of the queries corresponding to $D^{st}_{\mathcal{T}}(\mathcal{B})$.*

Proof Sketch. For proving this lemma, it is enough to prove the following four properties:

1. The change in the truth-value labelling of the query vertices of G_1 in lines 26-27 of Algorithm 8 corresponds to changing the belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ to $(D^{*}_{\mathcal{T}}(\mathcal{Q}_1, \mathcal{Q}_2)_1, \mathcal{Q}_2)$.

2. The change in the truth-value labelling of the query vertices of G_2 in line 29 of Algorithm 8 corresponds to changing the belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ to $(\mathcal{Q}_1, \mathcal{Q}_1)$.

3. The change in the truth-value labelling of the query vertices of G_2 in lines 31-32 of Algorithm 8 corresponds to changing the belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ to $(\mathcal{Q}_1, D^*_{\mathcal{T}}(\mathcal{Q}_1, \mathcal{Q}_2)_2)$.

4. Let $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3, \mathcal{Q}_4$ be DPWS's such that $\mathcal{Q}_3 \leq_K \mathcal{Q}_1 \leq_K \mathcal{Q}_4 \leq_K \mathcal{Q}_2$. If the query vertices are labelled $\mathbf{t}$ in correspondence with the distributed belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$, labelled $\mathbf{f}$ in correspondence with the distributed belief pair $(\mathcal{Q}_3, \mathcal{Q}_4)$, and labelled $\mathbf{u}$ otherwise, the resulting labelling corresponds to the distributed belief pair $(\mathcal{Q}_1, \mathcal{Q}_4)$.

Noting that a *says*-atom satisfied by the DPWS $\mathcal{Q}_1$ is true according to the distributed belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$, and that a *says*-atom satisfied by the DPWS $\mathcal{Q}_2$ is not false according to the distributed belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$, it is easy to see that Properties 2) and 4) hold.

Proof of Property 1): Suppose the labelling of query vertices corresponds to $(\mathcal{Q}_1, \mathcal{Q}_2)$. It is enough to show that lines 26-27 change a labelling of a query vertex $k : \varphi$ from u to t iff $(k \; says \; \varphi)^{D^*_{\mathcal{T}}(\mathcal{Q}_1, \mathcal{Q}_2)} = \mathbf{t}$.

So first suppose that line 26 changes the labelling of $k : \varphi$ from u to t. This means that there is an edge from $k : \varphi$ to a satisfied *says*-literal set vertex L. By Lemma 4.1, L fixes the truth value of φ to be true relative to T_k. Since the *says*-literal set vertex L is satisfied under the labelling of query vertices corresponding to $(\mathcal{Q}_1, \mathcal{Q}_2)$, this means that $\varphi^{(\mathcal{Q}_1, \mathcal{Q}_2), I} = \mathbf{t}$ for every structure I such that $(T_k)^{(\mathcal{Q}_1, \mathcal{Q}_2), I} \neq \mathbf{f}$. Since $(\mathcal{Q}_1, \mathcal{Q}_2) \leq_p D^*_{\mathcal{T}}(\mathcal{Q}_1, \mathcal{Q}_2)$, it follows that $\varphi^{D^*_{\mathcal{T}}(\mathcal{Q}_1, \mathcal{Q}_2), I} = \mathbf{t}$ for every structure I such that $(T_k)^{(\mathcal{Q}_1, \mathcal{Q}_2), I} \neq \mathbf{f}$. By Definition 4.34, $D^*_{\mathcal{T}}(\mathcal{Q}_1, \mathcal{Q}_2)_1 = (T_A)^{(\mathcal{Q}_1, \mathcal{Q}_2), I} \neq \mathbf{f}\})_{A \in \mathcal{A}}$. So $(k \; says \; \varphi)^{D^*_{\mathcal{T}}(\mathcal{Q}_1, \mathcal{Q}_2)} = \mathbf{t}$, as required.

Now suppose that line 27 changes the labelling of $k : \varphi$ from u to t. So there is a distributed belief pair $\mathcal{B}$ with $(\mathcal{Q}_1, \mathcal{Q}_2) \leq_p \mathcal{B} \leq_p D^*_{\mathcal{T}}(\mathcal{Q}_1, \mathcal{Q}_2)$ such that replacing *says* -atoms in φ by their truth values under $\mathcal{B}$ makes φ a tautology. In other

words, $\varphi^{\mathcal{B},I} = \mathbf{t}$ for any structure I, i.e. $\varphi^{D^*_{\mathcal{T}}(\mathcal{Q}_1,\mathcal{Q}_2),I} = \mathbf{t}$ for any structure I, i.e. $(k\ says\ \varphi)^{D^*_{\mathcal{T}}(\mathcal{Q}_1,\mathcal{Q}_2)} = \mathbf{t}$,as required.

Now suppose that $(k\ says\ \varphi)^{D^*_{\mathcal{T}}(\mathcal{Q}_1,\mathcal{Q}_2)} = \mathbf{t}$. Then $\varphi^{D^*_{\mathcal{T}}(\mathcal{Q}_1,\mathcal{Q}_2),I} = \mathbf{t}$ for every I such that $(T_k)^{(\mathcal{Q}_1,\mathcal{Q}_2),I} \neq \mathbf{f}$. Now there are two different cases: Either $\varphi^{\mathcal{Q}_1,\mathcal{Q}_2),I} = \mathbf{t}$ for every I such that $(T_k)^{(\mathcal{Q}_1,\mathcal{Q}_2),I} \neq \mathbf{f}$, or $\varphi^{D^*_{\mathcal{T}}(\mathcal{Q}_1,\mathcal{Q}_2),I} = \mathbf{t}$ for every structure I. In the first case, line 26 changes the labelling of $k : \varphi$ from u to t. In the second case, line 27 changes the labelling of $k : \varphi$ from u to t.

Property 3) can be proven in a similar way as property 1). □

The following theorem states that the result of the decision procedure is always in line with the well-founded semantics of dAEL(ID):

Theorem 4.1. *Let $\mathcal{T}$ be a distributed theory, let A be an agent, let let α be a dAEL(ID) formula. When A's theory T_A is queried about α, the decision procedure returns $(A\ says\ \alpha)^{wfm(T)}$, i.e. the truth value of $A\ says\ \alpha$ in the well-founded model of $\mathcal{T}$.*

Proof. Follows from Lemmas 4.3 and 4.2

□

4.4.7 Complexity of the decision procedure

The minimization of the information flow in the decision procedure is computationally expensive: The Query Minimization Procedure has a worst-case runtime that is exponential in the maximum of the number of different *says*-atoms in T, the size of the vocabulary Σ and the size of the domain D: Its **for**-loop has 3^n iterations, where n is the number of different *says*-atoms in T, and as min_incons has a worst-case runtime exponential in the maximum of the size of the vocabulary Σ and the size of the domain D. On the other hand, if we count each call to the Query Minimization Procedure as one step, the communication and loop-handling on has runtime quasilinear in the number of subqueries called.

To make the decision procedure practically applicable, a compromise between computational cost and limiting information flow will have to be found. One modification of the decision procedure that reduces the expected runtime, even though it does not reduce the worst-case runtime, is to not calculate the whole of $\mathbb{L}$ immediately in the Query Minimization Procedure, but to instead first calculate just one $L \in \mathbb{L}$, then do the communication necessary for determining whether this L actually makes the query true, and continue with the step-wise calculation of $\mathbb{L}$ only if the query has not yet been determined true.

4.5 Related Work

Most access control logics proposed in the literature have been defined in a proof-theoretical way, i.e. by specifying which axioms and inference rules they satisfy. This contrasts with Van Hertum et al.'s [VHCBD16] approach of defining dAEL(ID) semantically rather than proof-theoretically. This difference means that the tasks of defining decision procedures for these access control logics involve very different technical machinery.

Garg and Abadi [GA08, Gar08] and Genovese [Gen12] have defined Kripke semantics for many of the access control logics that were previously defined proof-theoretically in the literature. They introduced these Kripke semantics as a tool for defining decision procedures for those access control logics. Genovese [Gen12] follows the methodology of Negri and von Plato [NvP01, NvP11] of using a Kripke semantics of a modal logic to define Labelled Sequent Calculus, which forms the basis of a decision procedure for the logic.

Denecker et al. [DMT03] have defined a procedure for computing the well-founded model of an autoepistemic theory. This procedure might be extendable to a procedure for computing the well-founded model of dAEL(ID). However, such an extension of their procedure would not have the feature of minimizing the communication between

principals, and thus violate the need-to-know principle and cause privacy concerns (see section 4.4.3).

4.6 Conclusion

We have defined a query-based decision procedure for the well-founded semantics of dAEL(ID). When applying dAEL(ID) to access control, this decision procedure allows to determine access rights while minimizing the information flow between principals in order to enhance security and reduce privacy concerns.

Algorithm 8 $D^{st}_{\mathcal{T}}$-based decision procedure

Require: distributed theory $\mathcal{T}$, principal A, dAEL(ID) formula α
Ensure: truth-value $V \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$

$G :=$ the empty graph
add a new query vertex $A : \alpha$ to G
query_stack $:= \langle A : \alpha \rangle$
while query_stack $\neq \langle\rangle$ **do**
  $k : \varphi :=$ first element of query_stack
  $\mathbb{L} :=$ Query_Minimization_Procedure(T_k,φ)
  for $L \in \mathbb{L}$ **do**
    if G does not contain a *says*-literal set vertex L **then**
      add a new *says*-literal set vertex L to G
      for $l \in L$ **do**
        $k' :=$ the principal such that l is of the form k' *says* ψ or $\neg k'$ *says* ψ
        $\psi :=$ the formula such that l is of the form k' *says* ψ or $\neg k'$ *says* ψ
        if G does not contain a query vertex $k' : \psi$ **then**
          add a query vertex $k' : \psi$ to G
          add $k' : \psi$ to query_stack
        if l is k' *says* ψ **then**
          add to G a new edge labelled **t** from vertex L to vertex $k' : \psi$
        if l is $\neg k'$ *says* ψ **then**
          add to G a new edge labelled **f** from vertex L to vertex $k' : \psi$
    add to G a new edge from vertex $k : \varphi$ to vertex L
add the label **u** to all query vertices in G
finished $:= 0$
while finished $= 0$ **do**
  $G_1 := G$
  while in G_1 there is a query vertex labelled by **u** with an edge to a satisfied *says*-literal set vertex **do**
    change every **u**-label on a query vertex with an edge to a satisfied *says*-literal set vertex to **t**
    $G :=$ Make_labelling_of_query_vertices_good(G)
  $G_2 := G$
  change every **u**-label on a query vertex in G_2 to **f**
  while in G_2 there is a query vertex labelled by **f** with an edge to a potentially satisfied *says*-literal set vertex **do**
    change every **f**-label on a query vertex with an edge to a potentially satisfied *says*-literal set vertex to **u**
    $G :=$ Make_labelling_of_query_vertices_good(G)
  in G, change the label on all query vertices that are labelled **u** in G and labelled **t** in G_1 into **t**
  in G, change the label on all query vertices that are labelled **u** in G and labelled **f** in G_2 into **f**
  if no changes were made to G in the previous two lines **then**
    finished $:= 1$
$V :=$ the label on the query vertex $A : \alpha$ in G
return V

Chapter 5

Abstract Cumulative Aggregation

Abstract From any two conditional obligations "X if A" and "Y if B", cumulative aggregation derives the combined obligation "$X \cup Y$ if $A \cup (B \setminus X)$", whereas simple aggregation derives the obligation "$X \cup Y$ if $A \cup B$". We propose two types of systems, one consisting of factual detachment together with (simple) aggregation (FA systems), and the other consisting of factual detachment together with cumulative aggregation (FC systems). We give a representation result for FC systems, as well as for FA systems consisting of simple aggregation together with factual detachment. We relate FC and FA systems to each other and to input/output logics recently introduced by Parent and van der Torre.

5.1 Introduction

In this chapter, we contrast and study two different principles of aggregation for norms in the context of the framework of Abstract Normative Systems (ANS) due to Tosatto et al. [CTBvdTV12].

This one is intended as a general framework to compare logics for normative reasoning. Only fragments of the standard input/output logics [MvdT00] are covered by Tosatto et al., and so here we set ourselves the task of applying the framework to the input/output logic recently introduced by Parent and van der Torre [PvdT14a].

(Cf. also [PvdT14b].) Its most salient feature is the presence of a non-standard form of cumulative transitivity, called "aggregative" (ACT, for short). Such a rule is used in order to block the counter-examples usually given to the principle known as "deontic detachment": from the obligation of X and the obligation of Y if X, infer the obligation of Y.

Our contribution is first and foremost technical. We acknowledge that the benefits of using the theory of abstract normative systems may not be obvious to the reader. We will not discuss the question of whether it has a reasonable claim to be a general framework subsuming others, nor will we discuss the question of whether aggregative cumulative transitivity is, ultimately, the right form of transitivity.

A central feature of the Tosatto et al. account is that it abstracts away from the language of propositional logic. We recall that as initially conceived input/output logic is an attempt to generalize the study of conditional obligation from modal logic to the abstract study of conditional codes viewed as relations between Boolean formulas. The underlying language is taken from propositional logic. It contains truth-functional connectives, and is assumed to be closed under application of these connectives. It is natural to ask if one can extend the generality further, by working with an arbitrary language, viewed as a collection of items, and without requiring that the items under consideration be "given" or regimented in some special way. This level of abstraction favour the study of the relation among conditional norms without any demerit from the underlying logical language.

Tosatto et al.'s account has no apparatus for handling conjunction of outputs, and our main purpose in this chapter is to develop it to do so. We follow the ideas of so-called "multiple-conclusion logic", and treat normative consequence as a relation between sets, whose elements are understood conjunctively. No assumption about the inner structure of these elements is made.

An example of an abstract normative system studied in this chapter is given in Figure 5.1. It should be read as follows. Conditionals $A \to X, B \to Y, \ldots$ are the norms of the normative system. Each of A, X, B and Y is a set of language elements

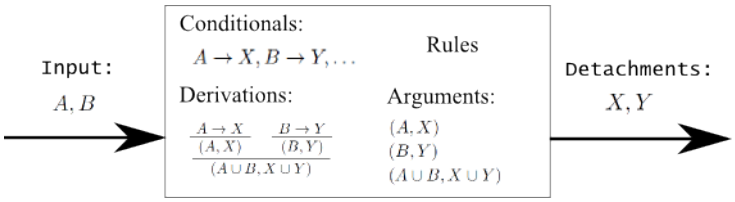


Figure 5.1: An Abstract Normative System

(whose inner structure remains unanalyzed). Sets are understood conjunctively on both sides of $\to$. The input I is a collection of language elements representing the context. Rules are used to generate derivations and arguments based on I. The set of detachments $\{X, Y, \ldots\}$ is the output consisting of all detached obligations. The elements of Figure 5.1 are explained in more detail in the next two sections.

The prime focus in [PvdT14a] was the contrast between two forms of transitivity, called "cumulative transitivity" and "aggregative cumulative transitivity". This chapter shifts the emphasis on the contrast between the following two forms of aggregation.

Simple aggregation If X is obligatory in context A, and Y is obligatory in context B, then $X \cup Y$ is obligatory in context $A \cup B$. In other words, simple aggregation derives the obligation "$X \cup Y$ if $A \cup B$" from any two conditional obligations "X if A" and "Y if B".[a]

Cumulative aggregation If X is obligatory in context A, and Y is obligatory in context B, then $X \cup Y$ is obligatory in context $A \cup (B \setminus X)$. In other words, cumulative aggregation derives the combined obligation "$X \cup Y$ if $A \cup (B \setminus X)$" from the same two conditional obligations.

The rule of simple aggregation gives the most straightforward way of collecting items as detachments are performed. When $A = B$, simple aggregation gives the rule "If X is obligatory given A, and Y is obligatory given A, then $X \cup Y$ is obligatory given

[a] Note that intersection as used in abstract normative systems does not correspond to disjunction in propositional logic. Take $(\{p\}, \{x\})$ and $(\{q\}, \{x\})$. The intersection of the two contexts yields $(\{\}, \{x\})$. Reasoning by cases would yield $(\{p \vee q\}, \{x\})$ instead.

A." A drawback of simple aggregation is that it does not capture transitive reasoning. Given the two conditional obligations "$\{x\}$ if $\{\}$" and "$\{y\}$ if $\{x\}$", simple aggregation only yields "$\{x, y\}$ if $\{x\}$". This motivates the rule of cumulative aggregation. In the particular case where $B = A \cup X$, cumulative aggregation yields the form of transitivity introduced by Parent and van der Torre [PvdT14a] under the name ACT. This is the rule $(A, X), (A \cup X, Y)/(A, X \cup Y)$. In our example, one gets "$\{x, y\}$ if $\{\}$."[b]

To summarize, we adress the following issues:

- How to develop the theory of abstract normative systems to handle conjunction of outputs and the form of cumulative transitivity described in [PvdT14a]?
- How to define the proof theory of the system? What are the most significant properties of the framework?
- How to provide a semantical characterisation, along with a representation result linking it with the proof theory?

The reminder of this chapter is organized as follows. In Section 5.2, we introduce some preliminary notions from deontic logic and its relations with aggregation. In Section 5.7 we provide a brief historical perspective on aggregation in the deontic literature. In Section 5.3, we provide a brief outline in regard to aggregation and a motivating example. In Section 5.4, we introduce FA systems for simple aggregation. In Section 5.5, we introduce FC systems for cumulative aggregation. We give representation results for both systems. In Section 5.6, we show how FA and FC systems relate to one another, and we discuss some properties of the systems. In Section 5.8 we show how FA and FC systems relate with the input/output logics introduced by Parent and van der Torre [PvdT14a]. In Section 5.9, we discuss possible extensions to the present framework as well as some limitations of the current approach. In Section 5.10, we provide a summary.

[b] As mentioned, it is not our purpose to discuss this rule in any greater depth. For more details on it, see Parent and van der Torre [PvdT14a].

5.2 Aggregation and Dilemmas

In its simplest form, aggregation is the derivation of the join obligation $\bigcirc(p \wedge q)$ from the obligations $\bigcirc(p)$ and $\bigcirc(q)$.

Aggregation is part of all deontic logics we are aware of.[c] Whereas there are many principles criticized and debated in deontic logic, aggregation has received less attention.

In this chapter we introduce a new kind of aggregation, called *Cumulative Aggregation*.

Deontic logic is intrinsically loaded with paradoxes. Thus we can distinguish two main lines of thought, Contrary-to-Duty (CTD) that studies and tries to tame the paradoxical or counterintuitive reasoning often generated by introducing violations in some way or another; and According-to-Duty (ATD) a more abiding perspective on which the main aim is to build frameworks resilient to counterintuitive or misleading detachments/inferences.

We also show how cumulative aggregation sheds a new light on some of the well known examples and paradoxes of deontic logic.

Most of the literature in deontic logic refers to CTD reasoning. From our point of view—and only focusing on the study of the aggregation—we follow ATD reasoning. ATD captures the behaviour of aggregation without many of the drawbacks that we could find in CTD. For example, most forms of aggregation in a CTD setting triggers what is known in the literature as "pragmatic oddity" [PS96].

In order to talk about CTD or ATD in a proper way, we need to distinguish between *primary* and *secondary* obligations. These concepts are linked to (and arise from) CTD reasoning. This distinction might seem arbitrary or artificial if it is not discussed in the context of CTD.

[c] For example, the deontic logics D and CD introduced by Chellas [Che80, Chap. 10] reject the principle of aggregation. In Chellas' terms, the axiom schemes OC and COC are not theorems in these logics.

A *primary* obligation describes some ideal behaviour, these obligations often come from modelling a particular problem or, in some sense, are given, e.g. unconditional obligations. A *secondary* obligation describes the current state of affairs; particularly in CTD, a secondary obligation is an obligation that comes into effect when the primary obligation is violated. Secondary obligations can be understood as subordinated to other obligations, that is primary.

Secondary obligations refer to an optimal choice under sub-ideal situations. In the sub-ideal situation that a primary obligation is violated, the best thing to do is to fulfil the secondary obligation.

As aforementioned we focus our study in ATD reasoning, but since CTD is the most prominent source for interesting examples in deontic logic, we borrow most of our examples from this line of reasoning. These examples are commonly constituted by some form of paradoxical construction i.e. Ross' Paradox [Ros44], Forrester's Paradox [For84] , the Good Samaritan Paradox [Pri58] and Chilsholm's Paradox [Chi63].

The following definitions formalize some relations between primary and secondary obligations:

Definition 5.1 (ATD [vdT97a])**.** *The conditional obligation* $\bigcirc(a/b)$ *is an According-to-Duty obligation of* $\bigcirc(c/d)$ *if and only if* c *logically implies* b*.*

Definition 5.2 (CTD [vdT97a])**.** *The conditional obligation* $\bigcirc(a/b)$ *is a Contrary-to-Duty obligation of* $\bigcirc(c/d)$ *if and only if* c *and* b *are inconsistent.*

Definition 5.3 (Dilemma [vdT97a])**.** *The conditional obligations* $\bigcirc(a/b)$ *and* $\bigcirc(c/d)$ *are a Dilemma if and only the obligations* a *and* c *are inconsistent.*

Figure 5.2 visualizes definitions of According-to-Duty, Contrary-to-Duty and Dilemmas.

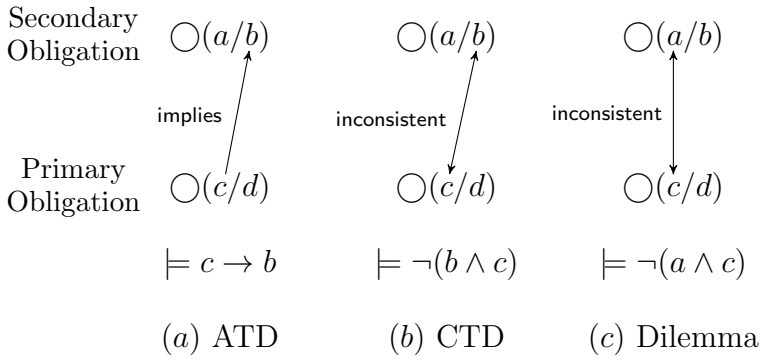


Figure 5.2: According-to-duty, Contrary-to-duty Obligations and Dilemmas

We use Chisholm's Paradox to exemplify these concepts.

Example 5.1 (Chilsholm's Paradox [Chi63])**.** *Consider the following set of conditional norms:*

(1) "You ought to assist your neighbour" $\bigcirc(a/\top)$

(2) "If you assist your neighbour, you should tell him that you will assist" $\bigcirc(t/a)$

(3) "If you do not assist your neighbour, you should not tell you will assist" $\bigcirc(\neg t/\neg a)$

(4) "You do not assist your neighbour" $\neg a$

We first identify primary and secondary obligations. The ideal situation is such that you help your neighbour and you tell him, thus (1) *constitute a primary obligation and the rest can be treated as secondary.*[d] *We could also argue that* (2) *is a primary obligation, we leave this discussion aside, since its not relevant at this point, and the example is only meant to exercise the concepts in Definitions 5.1 - 5.3.*

We have that,

- i) *the secondary obligation* $\bigcirc(t/a)$ *is according-to-duty of the primary obligation* $\bigcirc(a/\top)$ *since* a *logically implies* a*.*

[d] The rest of the sub-ideal situations can be ordered as follows: *1-* you help your neighbour, but you do not tell him before. 2- you do not help your neighbour (and you do not tell him). 3- you do not help your neighbour, but you tell him you will.

- ii) *the secondary obligation* $\bigcirc(\neg t/\neg a)$ *is contrary-to-duty of the primary obligation* $\bigcirc(a/\top)$ *since* $\neg a$ *and* a *are inconsistent.*

- iii) *the obligation* $\bigcirc(t/a)$ *and the obligation* $\bigcirc(\neg t/\neg a)$ *form are a dilemma since* t *and* $\neg t$ *are inconsistent.*

5.3 Motivating example

As discussed in Sections 5.1 and 5.2, aggregation and dilemmas are an intrinsic part of deontic logics. Whereas many principles have been extensively criticized and debated in deontic logic, aggregation has received less attention.

We first give an example to illustrate the rule of simple aggregation.

Example 5.2 (Double Taxation [Gre04])**.** *An extract of the Belgian-French bilateral agreement preventing double taxation. The agreement (among others) has two separate clauses:*

(i) A resident in Belgium should pay taxes in Belgium unless he/she is a French civil servant.

(ii) A resident in Belgium should pay taxes in Belgium unless he/she is working in France.

We can represent the obligation of β *given a context* α *as* $\bigcirc(\beta/\alpha)$ *[Che80]. Intuitively, we will use the following representation: "being resident in Belgium"* (r)*, "to be taxable"* (t)*, "being a French civil servant"* (fcs) *and "working in France"* (wf)*. We can consider this simple formulation for the example:*

(i) $\bigcirc(t/r \wedge \neg fcs)$

(ii) $\bigcirc(t/r \wedge \neg wf)$

Combining these two norms by means of simple aggregation gives us: $\bigcirc(t/r \wedge \neg fcs \wedge \neg wf)$. This is as it should be.

Now we give an example meant to illustrate the benefits of cumulative aggregation over simple aggregation.

Example 5.3 (Winter Chills)**.** *Suppose you are conformably sitting in your couch and you start to feel cold. Someone has opened the window to aerate the room. You wish to raise the temperature in the heating system, but first, you have to close the window. Intuitively, we will use the following representation: "the window is open" (wo), "the window is closed" (wc), "it is cold outside" (c) and "put the heating on" (h). The following two norms seem to apply:*

"If the window is open, then you ought to close the window" $\bigcirc(wc/wo)$

"if it is cold outside and the window is closed, then you ought to put the heating on" $\bigcirc(h/c \wedge wc)$

Cumulative aggregation gives $\bigcirc(wc \wedge h/wo \wedge c)$*. Intuitively: if the window is open and it is cold outside, then you ought to close the window and put the heating on. Simple aggregation gives* $\bigcirc(wc \wedge h/wo \wedge c \wedge wc)$*. Intuitively: if the window is open and closed, and it is cold outside, then you ought to close the window and put the heating on. The key difference is that the antecedent of the second obligation is contradictory.*

In what follows we introduce an abstract system to accommodate for simple aggregation. This system is meant to set grounds for the subsequent system for cumulative aggregation.

5.4 FA systems for simple aggregation

In this section, we introduce abstract normative systems for simple aggregation, and we give a representation result. Though FA systems may be interesting in their own right, in this chapter the main role of FA systems is to set the stage for FC systems for cumulative aggregation, introduced in the next section. Thus, although we talk

about normative systems and use examples from normative system it must be kept in mind that FA systems are not appropriate for all kinds of normative reasoning.

In general, a system $\langle L, C, R\rangle$ consists of a language L, a set of conditionals C defined over this language, and a set of rules R. The input is a set of sentences from L. A conditional $A \to X$ can be read as the norm "if A, then obligatory X". A normative system contains at least one set of norms, the regulative norms from which obligations and prohibitions can be detached. It may also contain permissive norms, from which explicit permissions can be detached, and constitutive norms, from which institutional facts can be detached [MvdT03, BvdT04, XvdT14]. In this chapter we do not consider permissive and constitutive norms. In the present setting, a system generates or produces a set of obligations.

All abstract normative systems we consider satisfy at least *factual detachment*. To represent factual detachment, we write (A, X) for the argument for X in context A, in other words, for input A the output contains X. Factual detachment is the rule $A \to X/(A, X)$, and says that if there is a conditional with the context as antecedent, then the output contains the consequent.

Besides factual detachment, FA systems have the rule of so-called simple aggregation. This one is usually given the form $(A, X), (A, Y)/(A, X \cup Y)$. In this chapter aggregation is given the more general form $(A, X), (B, Y)/(A \cup B, X \cup Y)$. This more general form allows for the inputs not to be the same. Given strengthening of the input, $(A, X)/(A \cup B, X)$, the two rules are equivalent. Since we do not assume strengthening of the input, our rule is strictly stronger.

Definition 5.4 (FA system with input)**.** *An FA* system *is a triple* $\langle L, C, R\rangle$ *with* L *a language,* $C \subseteq 2^L \times 2^L$ *a set of conditionals written as* $A \to X$*, and* R *a set of rules. For every conditional* $A \to X \in C$*,* A *and* X *are finite sets, where* R *consists of the rule of* factual detachment *(FD) and the rule of* aggregation *(AND):*

$$FD\ \frac{A \to X}{(A, X)} \qquad AND\ \frac{(A, X) \quad (B, Y)}{(A \cup B, X \cup Y)}$$

An input $I \subseteq L$ *for* system $\langle L, C, R\rangle$ *is a subset of the language.*

We write $a(A \to X) = A$ *for the* antecedent *of a conditional, and* $c(A \to X) = X$ *for the* consequent *of a conditional. We write* $a(C) = \cup\{a(A \to X) \mid A \to X \in C\}$ *for the union of the antecedents of all the conditionals in* C*. We write* $c(C) = \cup\{c(A \to X) \mid A \to X \in C\}$ *for the union of the consequents of all the conditionals in* C*.*[e]

The following example is meant to exercise the notation. We build a language, and introduce a set of conditionals and an input. The language L is the domain (or universe) of discourse. For the purpose of the example, L is a set of literals. Following Tosatto et al., we also introduce a complement function $\overline{e}$ for the elements e of the language L.

Example 5.4 (Sing and dance, adapted from Goble [Gob90])**.** *Given a language* L_0 *which does not contain formulas of the form* $\sim a$*, the language* L *is* $L_0 \cup \{\sim a \mid a \in L_0\}$ *where* $\sim$ *stands for classical negation. For* $a \in L$*, if* $a \in L_0$ *then* $\overline{a} = \sim a$*, and otherwise* $\overline{a} = b$ *for the* $b \in L_0$ *such that* $a = \sim b$*.*

Let L_0 *be* $\{x, y, d, s\}$*. Intuitively: "it is Spring" (*x*); "it is Sunday" (*y*); "a dance is performed" (*d*); and "a song is performed" (*s*).*

Suppose the conditionals $C_1 = \{y \to d, x \to s\}$ *apply to a wedding party. This says that on Sundays one ought to dance, and in Spring one ought to sing. The antecedents of the conditionals are:* $a(y \to d) = y$*;* $a(x \to s) = x$*;* $a(C_1) = \{x, y\}$*. Their consequents are:* $c(y \to d) = d$*;* $c(x \to s) = s$*;* $c(C_1) = \{s, d\}$*.*

We distinguish three related kinds of output from a system and an input, called derivations, arguments and detachments, respectively. A *derivation* is a finite tree, whose leaves are elements from the set of conditionals and whose root is a pair (A, X) obtained by successive applications of the rules, with the further constraint that $A \subseteq I$.[f] An *argument* is a pair (A, X) for which such a derivation exists, and X is a *detachment* if there exists an argument (A, X). Formally:

[e] To ease readability we will omit curly braces when referring to singleton sets, and we write $a \to x$ for $\{a\} \to \{x\}$.

[f] Alternatively, we could add the condition $A \subseteq I$ only to the definitions of arguments and detachments, or only to the definition of detachments. There are pros and cons to both choices. For

Definition 5.5 (Derivations `der`, Arguments *arg*, and Detachments *det*)**.** *Given a system* $\langle L, C, R\rangle$ *and an input* I*,*

- *a* derivation *of* (A, X) *on the basis of* I *in system* $\langle L, C\rangle$ *is a finite tree*[g] *using the rules* R*, with as leaves elements of* C*, and as root the pair* (A, X) *where* $A \subseteq I$ *and* $X \subseteq L$*.*

- *an* argument *is a pair* (A, X)*, such that there exists a derivation* d *with* $root(d) = (A, X)$*.*

- *a* detachment *is a set* X *such that there is an argument* (A, X)*.*

We write $\mathtt{der}(L, C, I, R)$ *for the set of all the derivations which can be constructed in this way, we write* $arg(L, C, I, R)$ *for the set of all such arguments, and we write* $det(L, C, I, R)$ *for the set of all such detachments.*

We write $leaves(d)$ *for the set of all the leaves of derivation* d*,* $i((A, X)) = A$ *for the input of a pair* (A, X) *and* $o((A, X)) = X$ *for the output of a pair* (A, X)*. Also we write* $i(D) = \cup\{i((A, X)) \mid (A, X) \in D\}$ *and* $o(D) = \cup\{o((A, X)) \mid (A, X) \in D\}$ *for the inputs and outputs of sets of such pairs.*

The essence of a derivation resides in the fact that they are always based on an input. This is reflected by the constraint $i(root(d)) \subseteq I$. But we stress that such a constraint is put on the root of the derivation only, and that all the other nodes need not verify this constraint. Otherwise we would not be able to chain conditionals together. Because of this, the property of closure under sub-derivations does not always hold. It depends on the rules being used. We will see an example of this phenomenon with system FC in Section 5.5. This also makes the proof of the representation theorem for FC trickier. The standard method of induction over the length of derivations is not available any more.

example, the advantage of our definition is that the set of derivations is smaller, but the disadvantage is that the set of derivations is not closed under sub-derivations, which complicates the proofs of the formal results.

[g] By a finite tree, we mean one with finitely many nodes.

A derivation is a relative notion, since it is meant to represent the inner structure of an argument. As argued before derivations are tied to the context giving a justification for the argument put forward based on what is, or is not, the case. In the literature, the notion of argument is defined in two ways. An argument is viewed as either a pair whose first element is a set of formulas (the support) and second element a formula (the conclusion), or as a derivation in a logical proof system, i.e. a sequence, tree or graph of logical formulas [PM12]. Here we choose the first definition. In the context of this study, the pair itself denotes a norm. However, it could represent any conditional statement. We use the term argument rather than norm, just to emphasize that we are interested in the relationship between a set of premises and its set of conclusions.

We now can briefly explain the notion of abstraction at stake in the theory of abstract normative systems. Intuitively, the detachment system treats the elements of L as atomic, in the sense that detachments have no relation with the logical structure of language L. Formally, we can replace one language L by another one L', define a one-to-one function f between elements of L and L', and extend f to subsets of L and C. Then we have $f(det(L, C, I, R)) = det(f(L), f(C), f(I), R)$. In this sense, it is an abstract theory.

We continue Example 5.4 to illustrate factual detachment and aggregation, as well as the distinction between derivations, arguments and detachments. In the absence of the rule of strengthening of the antecedent, one cannot derive that X is obligatory in context $A \cup B$ from the fact that X is obligatory in context A. This reflects the idea that arguments are minimal, in the sense that one cannot add irrelevant elements like B to their support. For example, if the input is $\{A, B\}$ and the sole conditional is $A \to X$, then there is no argument $(A \cup B, X)$. But X will be detached, since the input set triggers the conditional in question. The absence of the rule of strengthening of the antecedent does *not* reflect the fact that rules may leave room for exceptions.

Example 5.5 (Example 5.4 - Continued)**.** *Given* $L = L_0 \cup \{\sim a \mid a \in L_0\}$*, we say that an element* $a \in I$ *is a violation if there is a detachment containing* $\overline{a}$*, and this*

detachment is called a violated obligation. Moreover, we say that a detachment is a cue for action if it is not a violated obligation.

The derivations for $C_1 = \{y \to d, x \to s\}$ *and* $I_1 = \{x, y\}$ *are* $\mathtt{der}(L, C_1, I_1, FA) =$

$$\left\{ d_1 = \frac{y \to d}{(y,d)}\ FD\ ,\ d_2 = \frac{x \to s}{(x,s)}\ FD\ ,\ d_3 = \frac{\dfrac{x \to d}{(x,d)}\ FD \quad \dfrac{y \to s}{(y,s)}\ FD}{(\{x,y\},\{s,d\})}\ AND \right\},$$

the arguments are $arg(L, C_1, I_1, FA) = \{(y,d),(x,s),(\{x,y\},\{s,d\})\}$ *and the detachments are* $det(L, C_1, I_1, FA) = \{\{d\},\{s\},\{s,d\}\}$*, which are all cues for action. Thus* I_1 *does not contain violations. Factual detachment derives* d *and* s*, and aggregation combines them to* $\{d, s\}$*. First, note that some strengthening of the input is built in the aggregation inference rule AND, as we derive the conditional norm* $(\{x,y\},\{s,d\})$ *whose antecedent is stronger than the antecedent of the conditional norms in* C_1*. Second, note that, for the context where there is no singing* $I_2 = \{x, y, \bar{s}\}$*, we obtain exactly the same derivations, arguments and detachments. However, now* $\bar{s}$ *is a violation, and the detachments* $\{s\}$ *and* $\{s,d\}$ *are violated obligations, and only* $\{d\}$ *is a cue for action.*

Now consider $C_2 = \{\{x,y\} \to \{s,d\}\}$ *and, e.g.,* I_2*. The derivation is*

$$\mathtt{der}(L, C_2, I_2, FA) = \left\{ d_4 = \frac{\{x,y\} \to \{s,d\}}{(\{x,y\},\{s,d\})}\ FD \right\},$$

the arguments are $arg(L, C_2, I_2, FA) = \{(\{x,y\},\{s,d\})\}$ *and the detachments are* $det(L, C_2, I_2, FA) = \{\{s,d\}\}$*.*

It should not come as a surprise that the set of detachments is syntax-dependent. This follows at once from letting the rule of weakening of the output go. This phenomenon is familiar from the literature on belief revision.[h]

Theorem 5.1 gives a representation result for FA systems. The left-hand side of the bi-conditional pertains to the proof theory, while the right-hand side of it provides

[h] For more on the rule of weakening of the output, and the reason why it may be considered counter-intuitive, we refer the reader to the discussion in Goble [Gob90] (see also Parent and van der Torre [PvdT14a].)

a semantic characterization in terms of subset selection. For X to be derivable from a set of conditionals C on the basis of input I, X must be the union of the consequents of finitely many conditionals in C, which are all 'triggered' by the input set I.[i]

Theorem 5.1 (Representation result, FA)**.** *$X \in det(L, C, I, FA)$ if and only if there is some non-empty and finite $C' \subseteq C$ such that $a(C') \subseteq I$ and $X = c(C')$.*

Proof. The argument from the left to the right appeals to the following lemma.

Lemma 5.1. *For all $d \in \texttt{der}(L, C, I, FA)$,*

$$o(root(d)) = c(leaves(d)) \text{ and } i(root(d)) = a(leaves(d)) \qquad P(d)$$

Proof of Lemma 5.1. We prove the lemma by induction:

- Basis: the derivation d consists in an application of the rule FD; $d = \frac{A \to X}{(A,X)}$. Then, $root(d) = (A, X)$ and $leaves(d) = \{A \to X\}$. We have $o(root(d)) = c(leaves(d)) = X$ and $i(root(d)) = a(leaves(d)) = A$. Thus $P(d)$ holds.

- Inductive step: the derivation d ends with an application of the rule AND. That is,

$$d = \begin{array}{c} d_1 = \vdots \quad \vdots = d_2 \\ \hline (A,X) \quad (B,Y) \\ \hline (A \cup B, X \cup Y) \end{array} \text{AND}$$

  The sub-derivations d_1 and d_2 in d are such that $root(d_1) = (A, X)$ and $root(d_2) = (B, Y)$. The derivation d is such that $root(d) = (A \cup B, X \cup Y)$.

  As inductive hypothesis we assume that the claim holds for d_1 and d_2. That is, we assume that:

$$o(root(d_1)) = c(leaves(d_1)) \text{ and } i(root(d_1)) = a(leaves(d_1)) \qquad P(d_1)$$
$$o(root(d_2)) = c(leaves(d_2)) \text{ and } i(root(d_2)) = a(leaves(d_2)) \qquad P(d_2)$$

[i] In FA systems, we call 'triggered' those conditionals whose antecedents are in I.

We have the following chains of equalities:

$$\begin{aligned}
o(root(d)) &= X \cup Y \\
&= o(root(d_1)) \cup o(root(d_2)) \\
&= c(leaves(d_1)) \cup c(leaves(d_2)) \qquad \text{(by } P(d_1) \text{ and } P(d_2)\text{)} \\
&= c(leaves(d_1) \cup leaves(d_2)) \\
&= c(leaves(d))
\end{aligned}$$

$$\begin{aligned}
i(root(d)) &= A \cup B \\
&= i(root(d_1)) \cup i(root(d_2)) \\
&= a(leaves(d_1)) \cup a(leaves(d_2)) \qquad \text{(by } P(d_1) \text{ and } P(d_2)\text{)} \\
&= a(leaves(d_1) \cup leaves(d_2)) \\
&= a(leaves(d))
\end{aligned}$$

Hence, $P(d)$ holds under the assumption that $P(d_1)$ and $P(d_2)$ hold. □

With Lemma 5.1 in hand, we can establish the left-to-right direction of Theorem 5.1 as follows. Let $X \in det(L, C, I, FA)$. By definition 5.5, there is some derivation $d \in \texttt{der}(L, C, I, FA)$ such that $root(d) = (A, X)$ and $A \subseteq I$. We have $o(root(d)) = X$ and $i(root(d)) = A$. Consider $leaves(d)$. By construction, $leaves(d) \subseteq C$, $leaves(d) \neq \{\}$ and $leaves(d)$ is finite, since a derivation tree has finitely many nodes. Lemma 5.1 tells us that $o(root(d)) = c(leaves(d))$. Thus, $c(leaves(d)) = X$. Lemma 5.1 also tells us that $i(root(d)) = a(leaves(d))$, so that $a(leaves(d)) \subseteq I$ too. Thus, there exists some non-empty and finite $C' \subseteq C$ such that $a(C') \subseteq I$ and $X = c(C')$ as required.

We now establish the right-to-left direction of Theorem 5.1. Assume there is a non-empty and finite $C' \subseteq C$ such that $a(C') \subseteq I$ and $X = c(C')$. Let $C' = \{B_1 \to Y_1, \ldots, B_n \to Y_n\}$. We construct a derivation $d \in \texttt{der}(L, C, I, FA)$ such that $root(d) = (A, X)$ where $A = B_1 \cup \ldots \cup B_n$ and $X = Y_1 \cup \ldots \cup Y_n$.

$$\dfrac{\dfrac{\dfrac{\dfrac{B_1 \to Y_1}{(B_1, Y_1)}\,\text{FD} \quad \dfrac{B_2 \to Y_2}{(B_2, Y_2)}\,\text{FD}}{(B_1 \cup B_2, Y_1 \cup Y_2)}\,\text{AND} \quad \dfrac{B_3 \to Y_3}{(B_3, Y_3)}\,\text{FD}}{(B_1 \cup B_2 \cup B_3, Y_1 \cup Y_2 \cup Y_3)} \quad \dfrac{B_4 \to Y_4}{(B_4, Y_4)}\,\text{FD}}{\vdots \quad \text{AND}\ \dfrac{(B_1 \cup \ldots \cup B_{n-1}, Y_1 \cup \ldots \cup Y_{n-1}) \quad \text{FD}\ \dfrac{B_n \to Y_n}{(B_n, Y_n)}}{(B_1 \cup \ldots \cup B_n, Y_1 \cup \ldots \cup Y_n)}}$$

The fact that $d \in \texttt{der}(L, C, I, FA)$ is guaranteed by the fact that $a(C') = A \subseteq I$. Hence, $X \in det(L, C, I, FA)$ as required.

□

Corollary 5.1.1 (Monotonicity of *det*)**.** *$det(L, C, I, FA) \subseteq det(L, C', I, FA)$ whenever $C \subseteq C'$.*

The following example illustrates how to calculate the detachments using the semantic characterization described in the statement of Theorem 5.1.

Example 5.6 (Example 5.4 - Continued)**.** *We calculate $det(L, C_1, I_1, FA)$, now using Theorem 5.1. The set of conditionals C_1 has three non-empty subsets: $C_{1.1} = \{y \to d\}$, $C_{1.2} = \{x \to s\}$, and $C_{1.3} = \{y \to d, x \to s\}$. Here $a(C_{1.1}) \subseteq I_1$, $a(C_{1.2}) \subseteq I_1$ and $a(C_{1.3}) \subseteq I_1$. Also $c(C_{1.1}) = \{d\}$, $c(C_{1.2}) = \{s\}$ and $c(C_{1.3}) = \{s, d\}$. So $det(L, C_1, I_1, FA) = \{c(C_{1.1}), c(C_{1.2}), c(C_{1.3})\} = \{\{d\}, \{s\}, \{s, d\}\}$.*

5.5 FC systems for cumulative aggregation

In this section we introduce FC systems for cumulative aggregation. FC is much alike FA except that the rule of aggregation AND is replaced with that of cumulative aggregation CAND.

Definition 5.6 (FC system with input)**.** *An FC system is a triple $\langle L, C, R\rangle$ where R consists of the following rule of* factual detachment *(FD), and the rule of* cumulative aggregation *(CAND). We write $FC = \{FD, CAND\}$.*

$$FD = \frac{A \to X}{(A, X)} \qquad CAND = \frac{(A, X) \quad (B, Y)}{(A \cup (B \setminus X), X \cup Y)}$$

To illustrate the difference between FA and FC systems, we use the same example as the one that Parent and van der Torre [PvdT14a] use in order to motivate their rule ACT. We reckon that, compared to the framework described in [PvdT14a], the present framework does not yield any new insights into the analysis of the example itself.

Example 5.7 (adapted from Broome [Bro13])**.** *C contains two conditionals. One says that you ought to exercise hard everyday: $\{\} \to x$. The other says that, if you exercise hard everyday, you ought to eat heartily: $x \to h$. Intuitively, in context $\{\}$, we would like to be able to derive $\{x, h\}$, but not $\{h\}$.*

FA systems do not allow us to do it.

Let $I = \{\}$. With simple aggregation the set of derivations is $\mathtt{der}(L, C, I, FA) = \left\{d_1 = \frac{\{\} \to x}{(\{\}, x)}\ FD\right\}$, the set of arguments is $arg(L, C, I, FA) = \{(\{\}, x)\}$ and the set of detachments is $det(L, C, I, FA) = \{\{x\}\}$. Thus the desired obligation is not detached. Norms can be chained together only in so far as the input set contains their antecedent. Let $I' = \{x\}$. Then the set of derivations is $\mathtt{der}(L, C, I', FA) =$

$$\left\{ d_1 = \frac{\{\} \to x}{(\{\}, x)}\ FD \quad , \quad d_2 = \frac{x \to h}{(x, h)}\ FD \quad , \quad d_3 = \frac{\dfrac{\{\} \to x}{(\{\}, x)}\ FD \quad \dfrac{x \to h}{(x, h)}\ FD}{(x, \{x, h\})}\ AND \right\},$$

the set of arguments is $arg(L, C, I', FA) = \{(\{\}, x), (x, h), (x, \{x, h\})\}$ and the detachments are $det(L, C, I', FA) = \{\{x\}, \{h\}, \{x, h\}\}$.

With cumulative aggregation, the derivations for C and $I = \{\}$ are $\mathtt{der}(L, C, I, FC) =$

$$\left\{ d_1 = \frac{\{\} \to x}{(\{\}, x)}\ FD \quad , \quad d_2 = \frac{\dfrac{\{\} \to x}{(\{\}, x)}\ FD \quad \dfrac{x \to h}{(x, h)}\ FD}{(\{\}, \{x, h\})}\ CAND \right\}$$

The arguments are $arg(L, C, I, FC) = \{(\{\}, x), (\{\}, \{x, h\})\}$ and the detachments are $det(L, C, I, FC) = \{\{x\}, \{x, h\}\}$. Factual detachment allows us to detach $\{x\}$, and cumulative aggregation allows us to detach $\{x, h\}$ in addition. Like in [PvdT14a], h

C'	$f(C', D)$	$g(C', \{\})$
$\{\} \to x$	$\{x\}$	$\{D \mid x \in D\}$
$x \to h$	$\{\}$ if $x \notin D$, $\{h\}$ if $x \in D$	$\{D \mid x \notin D$ or $\{x, h\} \subseteq D\}$
$\{\} \to x$, $x \to h$	$\{x\}$ if $x \notin D$, $\{x, h\}$ if $x \in D$	$\{D \mid \{x, h\} \subseteq D\}$

Table 5.1: Functions f and g.

cannot be derived without x. Intuitively, the obligation to eat heartily no longer holds, if you take no exercise.

Definition 5.7 introduces the functions f and g, to be used later on in the semantic characterization of cumulative aggregation. Intuitively, given a set $D \subseteq L$, the function $f(C, D)$ gathers all the consequents of the conditionals in C that are triggered by D. The function $g(C, I)$ gathers all the sets D that extend the input set I and are closed under $f(C, \cdot)$.

Definition 5.7 (f and g)**.** *We define*

$$f(C, D) = \bigcup \{X \mid A \to X \in C; A \subseteq D\}$$

$$g(C, I) = \{D \mid I \subseteq D \supseteq f(C, D)\}$$

We illustrate the calculation of functions f and g continuing Example 5.7.

Example 5.8 (Example 5.7 - Continued)**.** *Consider Table 5.1. The left-most column shows the relevant subsets C' of C. The middle column shows what consequents can be detached depending on what set D is used as input. The right-most column shows the sets D extending I and closed under $f(C', D)$, for each subset C'.*

Theorem 5.2 gives a representation result for FC systems. For X to be derivable from a set of conditionals C on the basis of input I, X must be the union of the consequents of finitely many conditionals in C, which are either directly triggered by

the input set I (in the sense of Footnote *i*), or indirectly triggered by the input set I (via a chain of norms).

Theorem 5.2 (Representation result, FC)**.** *$X \in det(L, C, I, FC)$ if and only if there is some non-empty and finite $C' \subseteq C$ such that, for all $D \in g(C', I)$, we have $a(C') \subseteq D$ and $X = f(C', D)$.*

Proof. See [APv16][j] □

We show with an example how to calculate the detachments using the semantic characterization given in the statement of Theorem 5.2.

Example 5.9 (Example 5.8 - Continued)**.** *We again calculate $det(L, C, I, FC)$, now using Theorem 5.2. We use Table 5.1.*

The top row tells us that, $\{x\} \in det(L, C, I, FC)$. This is because, when $C' = \{\{\} \to x\}$ for all D in $g(C', \{\})$, $f(C', D) = \{x\}$.

The bottom row tells us that, $\{x, h\} \in det(L, C, I, FC)$. This is because, $C' = \{\{\} \to x, x \to h\}$ for all D in $g(C', \{\})$, $f(C', D) = \{x, h\}$.

We can also conclude that, $\{h\} \notin det(L, C, I, FC)$ because, for all C', there is a D in $g(C', \{\})$ such that $f(C', D) \neq \{h\}$.

Finally, the set of detachments is $det(L, C, I, FC) = \{\{x\}, \{x, h\}\}$.

5.6 Some properties of FA systems and FC systems

We start by showing how FA systems and FC systems relate to each other.

Definition 5.8 (Argument subsumption)**.** *Argument (A, X)* subsumes *argument (B, Y) if $A \subseteq B$ and $X = Y$. Given two sets of arguments S and T, we say that T subsumes S (notation: $S \sqsubseteq T$), if for all $(B, Y) \in S$ there is an argument $(A, X) \in T$ such that (A, X) subsumes (B, Y).*

[j] The proof of this theorem is credited to Xavier Parent.

Example 5.10. *Consider the following derivation.*

$$d = \frac{(A, X) \quad (A \cup B \cup X, X \cup Y)}{(A \cup B, X \cup Y)}\ CAND$$

The argument $(A \cup B, X \cup Y)$ *subsumes the argument* $(A \cup B \cup X, X \cup Y)$.

Proposition 4. $arg(L, C, I, FA) \sqsubseteq arg(L, C, I, FC)$.

Proof. Let $(A, X) \in arg(L, C, I, FA)$, where $A \subseteq I$. Let d be the derivation of (A, X) on the basis of I using the rules FD and AND. Let $leaves(d) = \{A_1 \rightarrow X_1, \ldots, A_n \rightarrow X_n\}$. We have $A = \bigcup_{i=1}^{n} A_i$ and $X = \bigcup_{i=1}^{n} X_i$.[k] That is,

$$(A, X) = (\bigcup_{i=1}^{n} A_i, \bigcup_{i=1}^{n} X_i)$$

One may transform d into a derivation d' of (A', X) on the basis of I using the rules FD and CAND. Keep the leaves and their parent nodes (obtained using FD) as they are in d, and replace any application of AND by an application of CAND. The result will be a tree whose root is

$$(A', X) = (A_1 \cup \bigcup_{i=2}^{n} (A_i \setminus \bigcup_{j=1}^{i-1} X_j), \bigcup_{i=1}^{n} X_i)$$

We have

$$A_1 \cup \bigcup_{i=2}^{n} (A_i \setminus \bigcup_{j=1}^{i-1} X_j) \subseteq \bigcup_{i=1}^{n} A_i \subseteq I \text{ and } \bigcup_{i=1}^{n} X_i = \bigcup_{i=1}^{n} X_i$$

On the one hand, $(A', X) \in arg(L, C, I, FC)$. On the other hand, (A', X) subsumes (A, X). □

Corollary 5.2.1. $det(L, C, I, FA) \subseteq det(L, C, I, FC)$

Proof. This follows at once from Proposition 4. □

We now point out a number of other properties of FA and FC systems.

[k] Strictly speaking, this follows from a lemma used in the proof of the representation result for FA systems, Lemma 5.1.

Proposition 5 (Applicability)**.** *The rules AND and CAND can be applied to any arguments* (A, X) *and* (B, Y)*.*

Proof. Trivial. Assume arguments (A, X) and (B, Y). By definition of an argument, $A \subseteq I$, $B \subseteq I$, $X \subseteq L$ and $Y \subseteq L$. Thus, $A \cup B \subseteq I$, $A \cup (B \setminus X) \subseteq I$ and $X \cup Y \subseteq L$. □

Proposition 6 (Premises permutation, FA)**.** *The result of applying AND to two arguments* (A, X) *and* (B, Y) *does not depend on the order of the two arguments.*

Proof. Straightforward. □

It is noteworthy that Proposition 6 fails for CAND, as shown by the following counterexample, where $A \neq B$:

$$\frac{(A,B) \qquad (B,A)}{(A, A \cup B)}\,\text{CAND} \quad \not\Longleftrightarrow \quad \frac{(B,A) \qquad (A,B)}{(B, A \cup B)}\,\text{CAND}$$

The arguments $(A, A \cup B)$ and $(B, A \cup B)$ are distinct.

Proposition 7 considers two successive applications of AND, or of CAND.

Proposition 7 (Associativity)**.** *Each of AND and CAND is associative, in the sense of being independent of the grouping of the pairs to which it is applied.*

Proof. The argument for AND is straightforward, and is omitted. For CAND, it suffices to show that the pairs appearing at the bottom of the following two derivations are equal:

$$\dfrac{\dfrac{\vdots}{(A,X)} \qquad \dfrac{\dfrac{\vdots}{(B,Y)} \qquad \dfrac{\vdots}{(C,Z)}}{(B \cup (C \setminus Y), Y \cup Z)}}{(A \cup ((B \cup (C \setminus Y)) \setminus X), X \cup Y \cup Z)} \qquad \dfrac{\dfrac{\dfrac{\vdots}{(A,X)} \qquad \dfrac{\vdots}{(B,Y)}}{(A \cup (B \setminus X), X \cup Y)} \qquad \dfrac{\vdots}{(C,Z)}}{(A \cup (B \setminus X) \cup (C \setminus (X \cup Y)), X \cup Y \cup Z)}$$

The fact that the two pairs in question are equal follows at once from the following two laws from set-theory:

$$(A \cup B) \setminus X = (A \setminus X) \cup (B \setminus X) \tag{5.1}$$

$$B \setminus (X \cup Y) = (B \setminus X) \setminus Y \tag{5.2}$$

We have:

$$\begin{aligned} A \cup ((B \cup (C \setminus Y)) \setminus X) &= A \cup (B \setminus X) \cup ((C \setminus Y) \setminus X) && \text{[by law (5.1)]} \\ &= A \cup (B \setminus X) \cup (C \setminus (X \cup Y)) && \text{[by law (5.2)]} \end{aligned}$$

□

Proposition 8. *FA systems are closed under sub-derivations in the following sense: given a derivation $d \in \mathtt{der}(L, C, I, FA)$, for all sub-derivations d' of d, $d' \in \mathtt{der}(L, C, I, FA)$–that is, $i(root(d')) \subseteq I$.*

Proof. Let $d \in \mathtt{der}(L, C, I, FA)$ with $root(d) = (A, X)$ and $A = A_1 \cup \ldots \cup A_n \subseteq I$ and $X = X_1 \cup \ldots \cup X_n$. Without loss of generality, we can assume that $n > 1$. By Proposition 7, d can be given the form:

$$\dfrac{\dfrac{\dfrac{\dfrac{A_1 \to X_1}{(A_1, X_1)}\,\text{FD} \quad \dfrac{A_2 \to X_2}{(A_2, X_2)}\,\text{FD}}{(A_1 \cup A_2, X_1 \cup X_2)}\,\text{AND} \quad \dfrac{\vdots}{(A_3, X_3)}\,\text{FD}}{(A_1 \cup A_2 \cup A_3, X_1 \cup X_2 \cup X_3)}\,\text{AND} \quad \vdots}{\vdots}$$

$$\text{AND}\ \dfrac{(A_1 \cup \ldots \cup A_{n-1}, X_1 \cup \ldots \cup X_{n-1}) \qquad \text{FD}\ \dfrac{A_n \to X_n}{(A_n, X_n)}}{(A_1 \cup \ldots \cup A_n, X_1 \cup \ldots \cup X_n)}$$

Let d' be a sub-derivation of d with root (A', X'). Clearly, $A' \subseteq A$, and so $A' \subseteq I$, since $A \subseteq I$. □

Proposition 9. *FC systems are not closed under sub-derivations.*

Proof. We prove this proposition by giving a counterexample. Let C be the set of conditionals $\{A \to X, X \to Y\}$ and let $I = \{A\}$. Consider the following derivation:

$$d = \dfrac{d_1 = \dfrac{A \to X}{(A, X)} \quad \dfrac{X \to Y}{(X, Y)}}{(A, X \cup Y)} = d_2$$

We have $i(root(d)) \subseteq I$, so that $d \in \texttt{der}(L, C, I, FC)$. Since $i(root(d_2)) = X$ and $X \not\subseteq I$, $d_2 \notin \texttt{der}(L, C, I, FC)$. □

Proposition 10 (Non-repetition)**.** *For every $d \in \texttt{der}(L, C, I, FA)$ with root (A, X) and leaves $leaves(d)$, there exists a derivation $d' \in \texttt{der}(L, C, I, FA)$ with the same root and the same set of leaves, such that each leaf in $leaves(d')$ is used at most once. The same holds for every derivation $d \in \texttt{der}(L, C, I, FC)$.*

Proof. We only consider the case of FC systems (the argument for FA systems is similar). Assume we have a derivation d with $root(d) = (A, X)$ and $leaves(d) = \{A_1 \to X_1, \ldots A_n \to X_n\}$. By Proposition 7, one can transform d into a derivation d' of the form

$$\dfrac{\dfrac{\dfrac{\dfrac{A_1 \to X_1}{(A_1, X_1)}\,\text{FD} \quad \dfrac{A_2 \to X_2}{(A_2, X_2)}\,\text{FD}}{\vdots\ \text{AND}} \quad \dfrac{A_3 \to X_3}{(A_3, X_3)}\,\text{FD}}{\vdots\ \text{AND}} \quad \vdots}{\vdots\ \text{AND}} \quad \text{FD}\ \dfrac{A_n \to X_n}{(A_n, X_n)}}{(A, X)}$$

Suppose that in d' some $A_l \to X_l$ decorates at least two distinct leaves. We show that we can eliminate the second one. To aid comprehension, let B be mnemonic for the following union, where $l \leq j$:

$$A_1 \cup (A_2 \setminus X_1) \cup (A_3 \setminus (X_1 \cup X_2)) \cup \ldots \cup (A_j \setminus (X_1 \cup \ldots \cup X_{j-1}))$$

Suppose we have the step:

$$\dfrac{\dfrac{\dfrac{\dfrac{A_1 \to X_1}{(A_1, X_1)} \quad \dfrac{A_2 \to X_2}{(A_2, X_2)}}{(A_1 \cup (A_2 \setminus X_1), X_1 \cup X_2)} \quad \dfrac{A_3 \to X_3}{(A_3, X_3)}}{(A_1 \cup (A_2 \setminus X_1) \cup (A_3 \setminus (X_1 \cup X_2)), X_1 \cup X_2 \cup X_3)} \quad \begin{array}{c} \vdots \\ \\ \end{array} \quad \begin{array}{c} \dfrac{A_j \to X_j}{(A_j, X_j)} \\ \vdots \\ \\ \end{array} \quad \dfrac{A_l \to X_l}{(A_l, X_l)}}{(B, \bigcup_{i=1}^{j} X_i)} \quad \dfrac{}{(B \cup (A_l \setminus \bigcup_{i=1}^{j} X_i), \bigcup_{i=1}^{j} X_i \cup X_l)}$$

where the sub-derivation with root $(B, \bigcup_{i=1}^{j} X_i)$ contains a leaf carrying $A_l \to X_l$. That is, $A_l \to X_l$ is one of $A_1 \to X_1$, ... and $A_j \to X_j$, and it is re-used immediately after $A_j \to X_j$. Since X_l is one of X_1, ... and X_j, $\bigcup_{i=1}^{j} X_i \cup X_l = \bigcup_{i=1}^{j} X_i$. On the other hand, $(A_l \setminus \bigcup_{i=1}^{j} X_i) \subseteq (A_l \setminus \bigcup_{i=1}^{l-1} X_i) \subseteq B$, so that $B \cup (A_l \setminus \bigcup_{i=1}^{j} X_i) = B$. Thus, we can remove from d' all the re-occurrences of the leaves as required. □

5.7 A Brief History of Aggregation

In this section we present a brief historical overview of the role of aggregation in deontic logic. The reader not interested in this brief addendum we recommend to skip this section.

We first give an overview of aggregation in the deontic logic literature. Most of the logical principles have been discussed and criticised exhaustively in the deontic literature, but aggregation has received less attention.

5.7.1 Aggregation in SDL

Standard Deontic Logic (SDL) was originally conceived by von Wright [vW51], although some changes have been made to von Wright's original system to achieve what is now known as SDL. Many of these changes have been adopted by von Wright himself, for example, one of the first revisions of 'the old system' can be found in [VW71].

SDL is defined by enriching propositional logic with the modal operator $\bigcirc$, the modal axiom schemata K and D, the *deontic necessitation rule* (and *modus ponens*

inherently present in propositional logic). As mentioned above, even though there are some differences in the definition we still refer to von Wright's original work.

Definition 5.9 (SDL [vW51])**.** *The set of formulae of $\mathscr{L}_{SDL}$ is inductively defined by the following EBNF rule:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \text{ , where } p \in \mathcal{P}$$

The symbols $\vee$, $\rightarrow$, $\leftrightarrow$, $\top$, and $\bot$ are treated is abbreviations in the standard way.

The modal formula $\bigcirc\varphi$ is to be read as "φ is obligatory".

Definition 5.10 (SDL System [Che80])**.** *The SDL system $\mathcal{S}_{\mathscr{L}_{SDL}}$ is characterized by the following axiom schemes and inference rules:*

- *All propositional tautologies.*
- *Inference rules:*

  $MP: \dfrac{\varphi \rightarrow \psi \qquad \varphi}{\psi}$ $\qquad$ $N: \dfrac{\vdash \varphi}{\bigcirc(\varphi)}$

- *Axiom Schemes:*

  $K : \bigcirc(\varphi \rightarrow \psi) \rightarrow (\bigcirc\varphi \rightarrow \bigcirc\psi)$

  $D : \neg(\bigcirc\varphi \wedge \bigcirc\neg\varphi)$

SDL is a normal modal KD-system following the standard Chellas' classification [Che80]. There exist alternative axiomatizations in the literature. The following alternative axiomatization, is also presented by Chellas [Che80].

In SDL, aggregation is represented by the scheme:

$$(\bigcirc\alpha \wedge \bigcirc\beta) \rightarrow \bigcirc(\alpha \wedge \beta)$$

This scheme is a theorem in SDL. We can clearly see this in SDL's alternative axiomatization.

Definition 5.11 (*SDL* Alternative System [Che80])**.** *The SDL system* $\mathcal{S}'_{\mathscr{L}_{SDL}}$ *is characterized by the following axiom schemes and inference rules:*

- *Inference rules:*

  $$ROM: \frac{\vdash \varphi \to \psi}{\bigcirc\varphi \to \bigcirc\psi}$$

- *Axiom Schemes:*

  $$OC : (\bigcirc\varphi \wedge \bigcirc\psi) \to \bigcirc(\varphi \wedge \psi)$$

  $$ON : \bigcirc\top$$

  $$OD : \neg \bigcirc \bot$$

The axiom scheme $D : \neg(\bigcirc\varphi \wedge \bigcirc\neg\varphi)$ is a theorem in any normal modal logic that also validates the axiom scheme $OD : \neg \bigcirc \bot$. Furthermore, these are (strongly) equivalent in such logics. In turn, OD's intuitive reading is to block impossible obligations rather than just blocking dilemmas. Moreover, under the presence of aggregation there is no way to distinguish D from OD.

Dilemmas arise naturally: it is not rare to find contradicting obligations. Since obligations can be generated by different parties, sources, etc, they may not be always coherent (or congruent). For example, you may be obliged to kill because it is your duty as serve in the army, but obligated not to kill because of personal and/or religious reasons ([Sar48]).

The following example adapted from [Hor93] illustrates how aggregation can be used to derive new obligations.

Example 5.11 (Alternative Service, adapted from [Hor93])**.** *Obligations may have many different sources. In this example, let us say that* (1) *is stated by your country's law, and* (2) *may come from your mother forbidding you to join the military. This can be seen as a re-interpretation of Sartre's student dilemma [Sar48].*

(1) "You ought to do either military service or alternative service" $\bigcirc(m \vee a)$

(2) "You ought not to do military service" $\bigcirc\neg m$

We can formalize this example as a Hilbert style derivation. A derivation is inductively defined as follows, every line in the derivation is either: (i) *an* assumption *from the set of obligations (or tautology);* (ii) *a statement resulting from the application of an inference rule.*

1- $\bigcirc(m \vee a)$ *Assum.*

2- $\bigcirc\neg m$ *Assum.*

3- $\bigcirc(a \wedge \neg m)$ *OC:1,2*

From the obligation to either do military or alternative service, and the obligation not to do military service, we conclude by using the OC inference rule, the obligation to do alternative service and not to do military service. This holds under the assumption of replacements of logical equivalents, since: $((a \vee m) \wedge \neg m) \equiv (a \wedge \neg m)$.

We can also formalize this example using a Gentzen style proof tree. We define proof trees inductively. The root of a proof tree is statement resulting from the application of an inference rule. The leafs are either: (i) *an* assumption *from the set of obligations (or tautology);* (ii) *the root of a proof tree.*

$$\frac{\bigcirc(s \vee m) \qquad \bigcirc\neg m}{\bigcirc(s \wedge \neg m)}\, OC$$

As we can see derivations and proofs trees are very similar. We will use one or the other depending on the context.

In the deontic logic literature, the validity of aggregation has been questioned in combination with the weakening principle, always in the context of dilemmas (i.e. see Van Fraassen [VF73], Goble [Gob13], Horty [Hor94]).

The *weakening principle* is embedded in many logical formalisms, i.e. monotonic logics. We will show later that this principle is one of the main culprits that–together with the aggregation rule–allows us to detach or infer any arbitrary formula from the existence of a dilemma.

There are several ways in which we can describe the weakening principle, for example, the most prominent in the literature is the following: if $\bigcirc\alpha$ and $\vdash \alpha \to \beta$, then $\bigcirc\beta$. In our case, for our present setting, we choose to represent this principle in the following way: $\dfrac{\bigcirc\alpha}{\bigcirc(\alpha \vee \beta)}\, W$ or $\bigcirc\alpha \to \bigcirc(\alpha \vee \beta)$ since $\vdash \alpha \to (\alpha \vee \beta)$.

Weakening can be used to derive simpler obligations from more complex obligations. For example, if we ought to pay (p) for a book and we ought to receive a receipt (r) due the payment, we can represent this as $\bigcirc(p \wedge r)$. Given the weakening principle, we can derive the ought to pay $\bigcirc(p)$. (For an extensive discussion on this topic see [Gob13]).

Any logic suited for deontic reasoning, that allows the existence of dilemmas, must not validate the principle of 'deontic explosion'. This principle can be formalized in SDL as follows:

$$DEX : (\bigcirc\alpha \wedge \bigcirc\neg\alpha) \to \bigcirc\beta$$

We alternatively present the proof theoretic formalization:

$$\frac{\bigcirc\alpha \qquad \bigcirc\neg\alpha}{\bigcirc\beta}\, DEX$$

Intuitively, the DEX principle states that in the existence of a dilemma, then everything is (or becomes) obligatory. Deontic dilemmas may naturally occur in within our set of obligations. Thus, deontic explosion must be rejected at once. This the main 'problem' raised by the introduction of dilemmas.

In SDL, the D axiom: $\neg(\bigcirc(\alpha) \wedge \bigcirc(\neg\alpha))$ serves as a constraint on the antecedent for DEX. Directly making dilemmas inconsistent by axiomatization. However, we still want a language (logic) that is expressive enough to allow us to reason without the absence of deontic dilemmas. We need to find a better way to deal with DEX, rather than completely excluding dilemmas from our logic formalism. Therefore, we need to find a different way to deal with DEX, while at the same time allowing for a rich amount of inferences that are valid (or desirable) for normative reasoning.

As a matter of clarification we provide the complete list of the schemes in the forthcoming quote.

(RM) If $\vdash A \to B$ then $\vdash \bigcirc A \to \bigcirc B$
(AND) $\vdash (\bigcirc A \wedge \bigcirc B) \to \bigcirc(A \wedge B)$
(EFQ) $\vdash (A \wedge \neg A) \to B$
(M) $\vdash \bigcirc(A \wedge B) \to (\bigcirc A \wedge \bigcirc B)$
(OR) $\vdash \bigcirc A \to \bigcirc(A \vee B)$
(RE) If $\vdash A \leftrightarrow B$ then $\vdash \bigcirc A \leftrightarrow \bigcirc B$

Lou Goble summarizes the subtle interaction between the different prospect culprits that lead to DEX:

> *The* problem *that deontic dilemmas present for deontic logic is simply the question of how to avoid deontic explosion, and (D), while at the same time accounting for the full range of inferences that do seem valid for normative concepts. Any logic that contains the rule (RM) [ROM] mentioned above and the aggregation principle ... and the principle of* ex falso quodlibet *... will* ipso facto *contain (DEX). Hence, to be adequate for deontic dilemmas, the logic must reject or restrict at least one of the principles (RM), (AND) and (EFQ) [ex falso quodlibet]. The question is, What is the best way to do that?.*
>
> *It is convenient to note that the inheritance rule (RM) is equivalent to the converse of (AND) and also to the principle (OR), namely ... given the rule of replacement for equivalents ... which seems a sine qua non for any reasonable deontic logic. That is to say, given (RM) then both (M) and (OR) are derivable (and, of course (RE)), and given either (M) or (OR), with (RE), then (RM) is derivable. Hence (M) and (OR) are equally implicated in the derivation of (DEX). [Gob05]*

There are many ways on which we could proceed. As aforementioned our choice is to drop the weakening principle. In Goble's terms this means the rejection of the

principle (OR). Thats not all, since this principle can be derived from the inheritance principle (RM). Thus, we also reject the principle of inheritance (RM).

Returning back to SDL, besides rejecting the axiom scheme D, we need to reject the axiom scheme K which–together with the necessitation inference rule–implies the derivation of DEX. Therefore we need to consider a non-normal modal logic (i.e. minimal models [Che80]). Non-normal modal logics are weaker normal modal logics, i.e. they allow fewer inferences to be valid. In non-normal modal logic, there we have a tradeoff between aggregation and weakening. Thus, the resulting logic only supports aggregation. In Chellas' terms, this is a logic of type **EC** [Che80]. In the following section we present such logic.

The main motivation for the weakening of the logic and the rejection of the weakening principle is solely to study aggregation in isolation.

For other other ways on how to prevent deontic explosion using non-monotonic logic techniques we refer the reader to [Hor93] and Two-Phase deontic logic [vdT97b, vdTT00].

5.8 Related research

We now compare our framework for cummulative aggregation with other I/O logic systems. As mentioned in Section 5.1, the present work extends the framework described by Tosatto et al. [CTBvdTV12] in order to handle conjunction of outputs along with the form of cumulative transitivity introduced by Parent and van der Torre [PvdT14a].

At the time, we are not able to report any formal result showing how the Tosatto et al. framework relates with the present one. Care should be taken here. On the one hand, the present account does not validate the rule of strengthening of the input, while the Tosatto et al. one does in the following restricted form: from $(\top, x)$, infer (y, x). On the other hand, in order to relate the proof-theory with the semantics, the

authors make a detour through the notion of deontic redundancy [vdT10]. A more detailed comparison between the two accounts is left as a topic for future research.

There are close similarities between the systems described in this chapter and the systems of I/O logic introduced by Parent and van der Torre [PvdT14a]. As explained in the introductory section, our rule CAND is the set-theoretical counterpart of their rule ACT. In both systems, weakening of the output goes away. At the same time there are also important differences between the two settings. First, the present setting remains neutral about the specific language to be used. It need not be the language of propositional logic. Second, the present account does not validate the rule of strengthening of the input.

Tosatto et al. explain how to instantiate their abstract normative system with propositional logic to obtain fragments of the standard input/output logics [MvdT00]. In this section we rerun the same exercise for the systems studied in [PvdT14a]. Unlike Tosatto et al., we argue semantically, and not proof-theoretically, because of the problem alluded to above: derivations in FC are not closed under sub-derivations.

For the reader's convenience, we first briefly recall the definitions of $\mathcal{O}_1$ and $\mathcal{O}_3$ given by Parent and van der Torre [PvdT14a]. Given a set X of formulas, and a set N of norms (viewed as pairs of formulas), $N(X)$ denotes the image of N under X, i.e., $N(X) = \{x : (a, x) \in N \text{ for some } a \in X\}$. $Cn(X)$ is the consequence set of X in classical propositional logic. And $x \dashv\vdash y$ is short for $x \vdash y$ and $y \vdash x$. We have $x \in \mathcal{O}_1(N, I)$ whenever there is some finite $M \subseteq N$ such that $M(Cn(I)) \neq \{\}$ and $x \dashv\vdash \bigwedge M(Cn(I))$. We have $x \in \mathcal{O}_3(N, I)$ if and only if there is some finite $M \subseteq N$ such that $M(Cn(I)) \neq \{\}$ and for all B, if $I \subseteq B = Cn(B) \supseteq M(B)$, then $x \dashv\vdash \wedge M(B)$.[l]

Theorem 5.3 (Instantiation)**.** *Let $\langle L, C, R\rangle$ be a FA system, or a FC system, with L the language of propositional logic (without $\top$) and C a set of conditionals whose*

[l] The proof-system corresponding to $\mathcal{O}_1$ has three rules: from (a, x) and $b \vdash a$, infer (b, x) (SI); from (a, x) and (a, y), infer $(a, x \wedge y)$ (AND); from (a, x) and $b \dashv\vdash a$, infer (b, x) (EQ). The proof-system corresponding to $\mathcal{O}_3$ may be obtained by replacing (AND) with (ACT). This is the rule: from (a, x) and $(a \wedge x, y)$, infer $(a, x \wedge y)$.

antecedents and consequents are singleton sets. Define $N = \{(a, x) \mid \{a\} \rightarrow \{x\} \in C\}$. *The following applies:*

i) If $X \in det(L, C, I, FA)$, *then* $\bigwedge X \in \mathcal{O}_1(N, I)$, *where* $\bigwedge X$ *is the conjunction of all the elements of* X;

ii) If $X \in det(L, C, I, FC)$, *then* $\bigwedge X \in \mathcal{O}_3(N, I)$.

Proof. See [APv16]. □

We recall the general definition of a system. A system is a tuple $\langle L, C, R\rangle$, where L is a language, C a set of conditional norms, and R is a set of rules. We also recall the rules to be used:

$$\text{FD} = \frac{A \rightarrow X}{(A, X)} \quad \text{SI} = \frac{(A, X)}{(A \cup B, X)} \quad \text{WO} = \frac{(A, X \cup Y)}{(A, X)} \quad \text{AND} = \frac{(A, X) \quad (B, Y)}{(A \cup B), X \cup Y)}$$

$$\text{CAND} = \frac{(A, X) \quad (B, Y)}{(A \cup (B \setminus X), X \cup Y)} \quad \text{CT} = \frac{(A, X) \quad (B, Y)}{(A \cup (B \setminus X), Y)}$$

In what follows we depict a set of possible systems that can be obtained by different combinations of such rules. We proceed in the following fashion. All the systems validate the rule of factual detachment, hence we take this minimalistic system as a start. Subsequently, we add the rules of aggregation (A), cumulative transitivity (X), and cumulative aggregation (C). Thus we obtain the systems FA, FX, and FC respectively. Later we add the rules of strengthening of the input (S), or weakening of the output (W) to the systems. Finally we add missing one out of S and W.

Figure 5.3 summarizes the relation between the systems depicted and those previously defined by Parent and van der Tore [PvdT14a] and Makinson and van der Torre [MvdT00].

Systems FA and FC are in accordance with $\mathcal{O}_1$ and $\mathcal{O}_3$ [PvdT14a] as stated in Theorem 5.3. To prove the claim that systems FASW and FCSW are equivalent to out_1 and out_3 respectively, and the logical implication between systems FCW and FXW, we appeal solely to the proof theoretic characterizations of the systems.

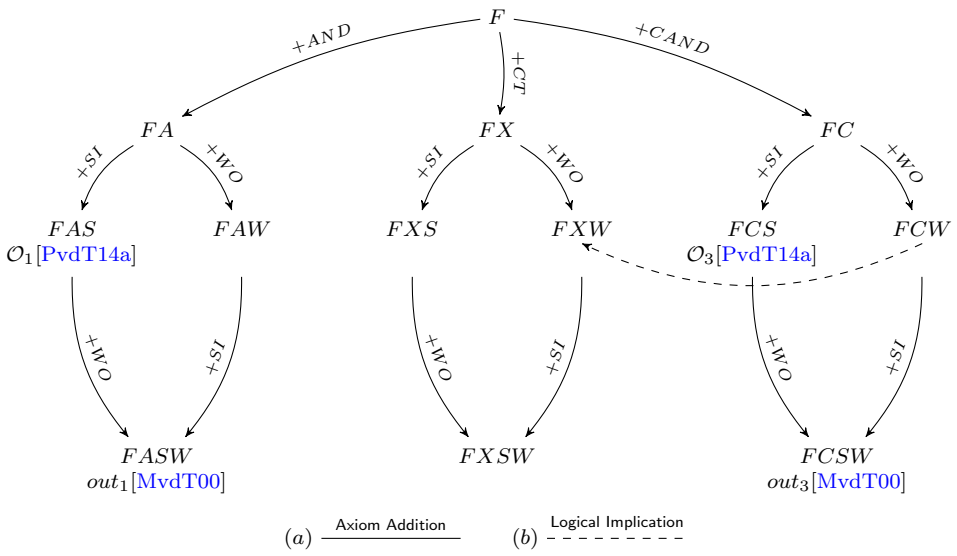


Figure 5.3: Relation between different systems

Proposition 11. $\{CAND, SI\} \Rightarrow AND$

Proof. Given arguments (A, X) and (B, Y) we build a derivation for $(A \cup B, X \cup Y)$ using the rules CAND and SI.

$$\dfrac{\dfrac{(A, X) \qquad (B, Y)}{(A \cup (B \setminus X), X \cup Y)}\,\text{CAND}}{(A \cup (B \setminus X) \cup X, X \cup Y)}\,\text{SI}$$

The consequent is equivalent to $(A \cup B, X \cup Y)$, as $(A \cup (B \setminus X) \cup X) = (A \cup B)$. □

Proposition 12. $\{CAND, WO\} \Rightarrow CT$.

Proof. Given arguments (A, X) and $(A \cup X, Y)$ we build a derivation for (A, Y) using the rules CAND and WO.

$$\dfrac{\dfrac{(A, X) \qquad (A \cup X, Y)}{(A \cup ((A \cup X) \setminus X), X \cup Y)}\,\text{CAND}}{(A, Y)}\,\text{WO}$$

□

The system FASW and out_1 [MvdT00] are characterised by the same set of rules. We briefly recall the definition of the proof system for out_1 called $deriv_1(G, A) = \{SI, AND, WO\}$, where G and A are analogous to the set of conditional norms C

and the input I respectively. We can have a glimpse on how these concepts relate in Theorem 5.3.

The same claim holds for system FCSW and out_3 [MvdT00], where $deriv_3(G, A) = \{SI, AND, CT, WO\}$. Even though these are not directly identical, Propositions 11 and 12 ensure that systems containing CAND, WO and SI also support AND and CT.

Finally, we prove the implication between systems FCW and FXW.

Proposition 13. *If* $X \in det(L, C, I, FXW)$, *then* $X \in det(L, C, I, FCW)$.

Proof. This follows at once from Proposition 12 □

It is noteworthy that the converse of Proposition 13 does not hold. It suffices to point out that in system FCW there is no rule that allows us to aggregate (or combine) consequents.

5.9 Discussion and Further Work

Section 5.8 briefly compares our present framework with other I/O logic systems. This comparison presupposed that strengthening of the input and weakening of the output had been added to FA and FC systems. In what follows we describe possible ways to modify the semantics in order to allow their addition.

Adding Strengthening of the Input. The natural way is to add the rule $(A, X)/(A \cup B, X)$ to the proof theory and keep the rest of the framework unchanged. Another way consists in reformulating the definition of derivation allowing for a subset of the antecedent to be included in the input set i.e. $A \cap I \neq \emptyset$. Regarding the representation results we conjecture that it suffices to relax the constraints in the relation between antecedents and input. That is, $A_i \subseteq a(C')$ and $A_i \subseteq I$ for Theorem 5.1. Given the subtleties of the closure functions and their interaction we refrain to

formulate any conjecture for Theorem 5.2. Thus we leave this analysis for further research.

Adding Weakening of the Output. Weakening of the output can be added in different ways. On the one hand we add the following rule to our proof theory $(A, X \cup Y)/(A, X)$. On the other hand we reformulate the definition of the notion of detachment in such a way that X is detached for input I if there is an argument (A, Y) such that $A \subseteq I$ and $X \subseteq Y$. Regarding the representation results we conjecture that it suffices to relax the constraints in the detachment set X from equality to set inclusion. That is, $X \subseteq c(C')$ and $X \subseteq f(C', D)$ for Theorems 5.1 and 5.2 respectively. It still remain to be seen if the proof of the representation results still goes trough, and in what form.

There are other rules that one may want to add. These include: plain transitivity, $(A, X), (X, Y)/(A, Y)$, and reasoning by cases, $(A \cup \{a\}, X), (B \cup \{\bar{a}\}, Y)/(A \cup B, X \cup Y)$. We leave the formal analysis of such kinds of extensions as a topic for further research.

We end this section by discussing two problems.

Irrelevant Obligations. This problem was pointed out by Stolpe [Sto08] in the context of a discussion of the principle of transitivity. It affects Makinson/van der Torre's initial framework and Parent and van der Torre [PvdT14a]'s one. Roughly speaking, the irrelevant obligation problem can be described as follows. In a violation context, i.e. $\bar{a} \in I$ and $a \in X$, one can detach the obligation of any state of affairs as long as it appears as consequent of a conditional. In our notation, this means that given the set of conditionals $C = \{\{\} \rightarrow \bar{x}, b \rightarrow y\}$ and input $I = \{x\}$, $y \in det(L, C, I, R)$. This looks counter-intuitive. We do not have this problem in either system FA or system FC.

FA system. Look at the statement of Theorem 5.1. There is only one candidate $C' \subseteq C$ meeting the requirement $a(C') \subseteq I$. This is $C' = \{\{\} \to \bar{x}\}$. Hence according to Theorem 5.1, $det(L, C, I, FA) = \{\{\bar{x}\}\}$.

FC system. We use a similar table as Table 5.1 for Example 5.8:

C'	$f(C', D)$	$g(C', \{x\})$
$\{\} \to \bar{x}$	$\{\bar{x}\}$	$\{D \mid \{x, \bar{x}\} \in D\}$
$b \to y$	$\{\}$ if $b \notin D$, $\{y\}$ if $b \in D$	$\{D \mid (x \in D \text{ and } b \notin D) \text{ or } \{x, b, y\} \subseteq D\}$
$\{\} \to \bar{x}$, $b \to y$	$\{\bar{x}\}$ if $b \notin D$, $\{\bar{x}, y\}$ if $b \in D$	$\{D \mid (\{x, \bar{x}\} \in D \text{ and } b \notin D) \text{ or } \{x, \bar{x}, b, y\} \subseteq D\}$

There is only one non-empty $C' \subseteq C$ fulfilling the requirement "for all $D \in g(C', I)$, $a(C') \subseteq D$" mentioned in the statement of Theorem 5.2. This is $C' = \{\{\} \to \bar{x}\}$. For this C', $f(C', D) = \{\bar{x}\}$, so that $det(L, C, I, FC) = \{\{\bar{x}\}\}$.

Pragmatic Oddity. The derivation below depicts the problem known as of *Pragmatic Oddity* [PS96]. We use the *dog and sign scenario*, the letters d and s stand for "there is a dog" and "there is a warning sign" respectively. The intended reading is the following, "there should not be a dog" $(\{\}, \bar{d})$, and, "if there is a dog, then it should be a warning sign" (d, s).

$$\frac{(\{\}, \bar{d}) \qquad (d, s)}{(d, \{\bar{d}, s\})}\ \text{AND}$$

Given the fact that we indeed have a dog (i.e. $d \in I$) we can conclude that we should not have a dog and we should have a warning sign. This conclusion might seem unnatural a first glance, but it is possible to find examples on which these type of derivation makes sense. For example, if you do not pay the rent, you still ought to pay the rent, plus some fine regulated for the delay.

There are several courses of action we could take in order to forbid or restrict these kind of derivations. The natural way is to restrict the application of the rule

CAND. We allow the application of the rule only if for every $a \in A \cup (B \setminus X)$ there is no $\bar{a} \in X \cup Y$. Another approach is to directly constraint the notions of either derivations or arguments (Definition 5.5) to exclude trees with roots (resp. pairs) in which the antecedent and consequent contain complementary elements. All the proposed solutions are of proof theoretical nature. It remains to be seen which semantics will follow from this constraints.

5.10 Conclusion

We have extended the Tosatto et al. framework of abstract normative systems in order to handle conjunction of outputs along with the aggregative form of cumulative transitivity introduced by Parent and van der Torre [PvdT14a]. We have introduced two abstract normative systems, the FA and FC systems. We have illustrated these two systems with examples from literature, and presented two representation theorems for these systems. We have also shown how they relate to the original I/O systems.

FA systems. They supplement factual detachment with the rule of simple aggregation, taking unions of inputs and outputs. The representation theorem Theorem 5.1 shows that the sets of formulas that can be detached in FA precisely correspond to sets of conditionals that generate this output.

FC systems. They supplement factual detachment with the rule of cumulative aggregation, a subtle kind of transitivity or reuse of the output, as introduced in [PvdT14a]. The representation theorem Theorem 5.2 shows how the cumulative aggregation rule corresponds to the reuse of the detached formulas.

Chapter 6

Conclusion and Further Work

Abstract. In this chapter we provide a recap of the research questions and what conclusions we draw after answering them. We later provide some discussion and directions for future lines of research.

6.1 Conclusion

We have described how logical formalisms can be employed in various ways in relation with access control. We now recapitulate over the research questions presented in Section 1.2 and discuss how they have been answered.

Research Question 1. *How to formally characterize the different reasons for revoking or denying access rights?*

We have studied the reasons that principals may have to perform revocations.

We have developed Trust Delegation Logic (TDL) in order to formalize the reasons for revocating. We have provided different scenarios to motivate the reasons for revocating and their relation with TDL. TDL incorporates two epistemic operators that together allowed us to define different levels of trust among principals. Accordingly, TDL was also equipped with a modal operator that allows principals to make public announcements about statements describing their access control policy.

Finally, the non-monotonic behaviour inherent to delegation revocation was captured including negation by failure to TDL.

Research Question 2. *Can the different revocation schemes be defined in such a way that they are in line with the different reasons for revocating?*

First and foremost we have tackled the problems encountered in Hagström et al.'s [HJPPW01] revocation framework. We developed a refined framework with clear, formal graph-theoretic definitions for each revocation scheme proposed/considered. We have formally studied the relation between the reasons for revocating and the definitions of revocations schemes. TDL allowed us to formulate a postulate describing the desired behaviour, following the axiomatic method. The proposed postulate describes the correspondence between the behaviour of the revocation schemes and the reasons for revocating, and ensures that access is granted whenever this is justifiable on the basis of the reasons for granting and revocating.

Research Question 3. *How can a says-based access control logic support denials in a natural way?*

We have studied distributed autoepistemic logic (dAEL(ID)) and its application to access control. We have proposed dAEL(ID) with well-founded semantics as a characterization of the *says* operator. As previously discussed, issuing denials may cause a non-monotonic behaviour in which granting new access rights does not necessarily result in a principal having more access rights. dAEL(ID) is a non-monotonic logic, thus allowing to naturally capture the behaviour required for a issuing such denials. This (new) non-monotonic perspective on the *says* operator sheds light over its intended behaviour allowing for new semantics to be defined.

Research Question 4. *Can a decision procedure for such an access control logic be developed?*

We have defined a query-based decision procedure for the well-founded semantics of dAEL(ID). The existence of a decision procedure ensures that dAEL(ID) is decidable. The decision procedure allows to determine access rights with a minimal

amount of communication between the involved principals, thus minimizing the information flow in the system. To achieve this, first our decision procedure performs meta-reasoning about the local policies relative to each principal. Second the decision procedure handles the communication among principals. When loops are formed, these are detected and treated in accordance with dAEL(ID)'s well-founded semantics.

We conclude in a general note that non-monotonicity is a core property required for the specification of access control policies in systems that aim to support denial of rights. However it is worth noticing that such logics with a high level of expressivity have a high computational cost.

6.2 Further Work

We first have studied revocation for a version of delegation that does not have any bound on the length of delegation chains. However, TDL lends itself very well also to delegation with such a bound. Indeed, for this purpose a somewhat reduced version of TDL, which lacks the $T_i\mathbb{D}$ and $T_i\mathbb{S}$ trust operators, would be sufficient. It would be interesting to study the possibility to define a systematic revocation framework for such bounded delegation that satisfies a desirable property analogous to the one that our framework for revoking unbounded delegation has been shown to satisfy.

Some reasons for revoking rights cannot be captured in TDL: User A may revoke B's rights just because she does not consider it to be useful for B to have the right in question, even though she strongly trusts B. And A may choose a certain kind of revocation scheme based on considerations of responsibility, not just considerations of trust. Our preliminary investigations into this topic suggest that our refined framework also corresponds well to these not trust-based reasons for revocating, but further investigations are due in order to develop an extension of TDL that can formalize such reasons and proof our system to satisfy a desirable property based on this extended logic.

In order to develop an axiomatic theory of revocation schemes similar to the application of the axiomatic method in social choice theory, other desirable properties of revocation schems or revocation frameworks need to be identified and compared to the desirable property that we proposed.

We propose two main extensions for TDL logic: (i) extending the logic TDL with the concepts of utility and responsibility; and (ii) studying the semantics for TDL logic. First the main motivation for (i) is to be able to talk about if and when a principal 'needs' to have a right and how this affects the revocation of rights.

Second, the motivation for (ii) is to improve the theoretic study of the logic TDL, allowing us to study the relation between design decisions for the logic and the revocation frameworks characterised by the various possible logics from different points of view rather than being forced to solve this in a proof theoretic realm. We propose to apply the methodology used for dAEL(ID)'s semantics, i.e. AFT, in order to capture TDL's non-monotonic behaviour.

Although we tried to stay close to Hagström's framework as possible, it would be of interest to study what happens if one takes into account other approaches to revocation.

Given that our decision procedure for dAEL(ID) has in the worst case a exponential runtime (see Section 4.4.7), a more efficient decision procedure will have to be developed for dAEL(ID) or an expressively rich subset of it in order to apply it in practice. One of the reasons why our decision procedure is so inefficient is that the minimization of information flow is very expensive. Instead of the minimizing information flow to an absolute minimum as our decision procedure does, a practically applicable decision procedure would have to find a compromise between computational cost and limiting information flow. Nevertheless, we consider our ideal minimization of information flow an interesting proof of concept as a foundation for further research.

Our decision procedure aims at proving a query in terms of queries to other principals. In this process, it cautiously handles possible loops between such queries.

This is highly reminiscent of the way *justifications* are defined, for instance for logic programs [DBS15]. Hence it may be interesting to define justification semantics for dAEL(ID).

Another way to characterize dAEL(ID) would be through an axiomatic proof system. Most other access control logics are defined in this way, hence this would make dAEL(ID) more straightforwardly comparable to other access control logics. To the best of our knowledge, no proof system has been developed for autoepistemic logic with the well-founded semantics.

The possibility of defining an intuitionistic variant of dAEL(ID) should be studied. This will allow for a direct comparison with other *says*-based approaches common in the literature. Furthermore, it is worthwhile to study the possibility of adding features of other state-of-the-art access control logics to dAEL(ID), e.g. the support for explicit time and system state incorporated in BL [Gar09, GP12].

Bibliography

Bibliography

[Aba08] Martín Abadi. Variations in Access Control Logic. In *9th International Conference on Deontic Logic in Computer Science*, pages 96–109, 2008. 2

[ABB+11] Guillaume Aucher, Steve Barker, Guido Boella, Valerio Genovese, and Leendert van der Torre. Dynamics in Delegation and Revocation Schemes: A Logical Approach. In Yingjiu Li, editor, *Data and Applications Security and Privacy XXV*, volume 6818 of *Lecture Notes in Computer Science*, pages 90–105. Springer Berlin, 2011. 42, 44

[AC17] Diego Agustín Ambrossio and Marcos Cramer. A Query-Driven Decision Procedure for Distributed Autoepistemic Logic with Inductive Definitions. In *Workshop on Foundations of Computer Security (FCS) 2017*, 2017. 9

[AF99] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *Proceedings of the 6th ACM conference on Computer and communications security* , pages 52–62. ACM, 1999. 2

[APv16] D. A. Ambrossio, X. Parent, and L. van der Torre. A representation theorem for abstract cumulative aggregation. Technical report, University of Luxembourg, 2016. 135, 148

[AXL16] Diego Agustín Ambrossio, Parent Xavier, and van der Torre Leon. Cumulative aggregation. In *Deontic Logic and Normative Systems 13th International Conference*, pages 1–15, 2016. 9

[BJS96] Elisa Bertino, Sushil Jajodia, and Pierangela Samarati. A Non-timestamped Authorization Model for Data Management Systems. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, CCS '96, pages 169–178, New York, NY, USA, 1996. ACM. 3, 22

[Bro13] J. Broome. *Rationality Through Reasoning*. Wiley-Blackwell, 2013. 133

[BSJ97] E. Bertino, P. Samarati, and S. Jajodia. An extended authorization model for relational databases. *Knowledge and Data Engineering, IEEE Transactions on*, 9(1):85–101, Jan 1997. 3, 22

[BvdT04] Guido Boella and Leendert van der Torre. Regulative and constitutive norms in normative multiagent systems. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR 2004)*, pages 255–266, 2004. 125

[CAV15] Marcos Cramer, Diego Agustín Ambrossio, and Pieter Van Hertum. A Logic of Trust for Reasoning about Delegation and Revocation. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies* , pages 173–184. ACM, 2015. 9

[Che80] B. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980. 120, 123, 141, 142, 146

[Chi63] R. Chisholm. Contrary-to-duty imperatives and deontic logic. *Analyse*, 24:33–36, 1963. 121, 122

[CTBvdTV12] S. Colombo Tosatto, G. Boella, L. van der Torre, and S. Villata. Abstract normative systems: Semantics and proof theory. 2012. 116, 146

[DBBD14a] Broes De Cat, Bart Bogaerts, Maurice Bruynooghe, and Marc Denecker. Predicate Logic as a Modelling Language: The IDP System. *CoRR*, abs/1401.6312, 2014. 14

[DBBD14b] Broes De Cat, Bart Bogaerts, Maurice Bruynooghe, and Marc Denecker. Predicate Logic as a Modelling Language: The IDP System. *CoRR*, abs/1401.6312, 2014. 64

[DBS15] Marc Denecker, Gerhard Brewka, and Hannes Strass. A Formal Theory of Justifications. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, volume 9345 of *Lecture Notes in Computer Science*, pages 250–264. Springer, 2015. 158

[Dem04] Robert Demolombe. Reasonig about trust: A formal logical framework. In Christian Jensen, Stefan Poslad, and Theo Dimitrakos, editors, *Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 291–303. 2004. 42, 44, 45, 49

[Den98] Marc Denecker. The Well-Founded Semantics Is the Principle of Inductive Definition. In Jürgen Dix, Luis Fariñas del Cerro, and Ulrich Furbach, editors, *JELIA*, volume 1489 of *LNCS*, pages 1–16. Springer, 1998. 31, 35, 36, 37

[Den00] Marc Denecker. Extending Classical Logic with Inductive Definitions. In John W. Lloyd, Verónica Dahl, Ulrich Furbach, Manfred Kerber,

Kung-Kiu Lau, Catuscia Palamidessi, Luís Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *CL*, volume 1861 of *LNCS*, pages 703–717. Springer, 2000. 64

[DMT98] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Fixpoint 3-valued semantics for autoepistemic logic. In *AAAI'98*, pages 840–845, Madison, Wisconsin, July 26-30 1998. MIT Press. 73

[DMT00] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, volume 597 of *The Springer International Series in Engineering and Computer Science*, pages 127–144. Springer US, 2000. 18, 72

[DMT03] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Uniform semantic treatment of default and autoepistemic logics. *Artif. Intell.*, 143(1):79–122, 2003. 73, 74, 75, 113

[DMT04] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation*, 192(1):84–121, July 2004. 20, 70, 73, 77

[DMT11] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Reiter's Default Logic Is a Logic of Autoepistemic Reasoning And a Good One, Too. In Gerd Brewka, Victor Marek, and Mirosław Truszczyński, editors, *Nonmonotonic Reasoning – Essays Celebrating Its 30th Anniversary*, pages 111–144. College Publications, 2011. 64

[DV14] Marc Denecker and Joost Vennekens. The Well-Founded Semantics Is the Principle of Inductive Definition, Revisited. In Chitta Baral,

Giuseppe De Giacomo, and Thomas Eiter, editors, *KR*, pages 1–10. AAAI Press, 2014. 64, 86, 87

[DWD11] Stef De Pooter, Johan Wittocx, and Marc Denecker. A prototype of a knowledge-based programming environment. *CoRR*, abs/1108.5667, 2011. 15

[Fag78] Ronald Fagin. On an Authorization Mechanism. *ACM Trans. Database Syst.*, 3(3):310–319, September 1978. 3, 22

[For84] James William Forrester. Gentle murder, or the adverbial samaritan. *Journal of Philosophy*, 81:193–196, 1984. 121

[FSG+01] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001. 2

[GA08] Deepak Garg and Martín Abadi. *Foundations of Software Science and Computational Structures: 11th International Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings*, chapter A Modal Deconstruction of Access Control Logics, pages 216–230. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. 113

[Gar08] Deepak Garg. Principal-Centric Reasoning in Constructive Authorization Logic, 2008. 113

[Gar09] Deepak Garg. *Proof Theory for Authorization Logic and Its Application to a Practical File System*. PhD thesis, 2009. 158

[Gen12] Valerio Genovese. *Modalities in Access Control: Logics, Proof-theory and Application*. PhD thesis, 2012. 2, 113

[Gin88] Matthew L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4(3):265–316, 1988. 20

[Gob90] Lou Goble. A logic of good, should, and would. I. *J. Philos. Log.*, 19(2):169–199, 1990. 126, 129

[Gob05] Lou Goble. A logic for deontic dilemmas. *Journal of Applied Logic*, 3(3–4):461 – 483, 2005. {DEON} 04The 7th international workshop on the uses of Deontic Logic in Computer Science. 145

[Gob13] Lou Goble. Prima facie norms, normative conflicts and dilemmas. In Dov Gabbay, Jeff Horty, Xavier Parent, Ron van der Meyden, and Leon van der Torre, editors, *Handbook of Deontic logic and Normative Systems*, chapter 4, pages 241–352. College Publications, 2013. 143, 144

[GP12] Deepak Garg and Frank Pfenning. Stateful Authorization Logic – Proof Theory and a Case Study. *Journal of Computer Security*, 20(4):353–391, 2012. 2, 63, 158

[Gre04] Eric Gregoire. Fusing legal knowledge. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE InternationalConference on*, pages 522–529. IEEE, 2004. 123

[GW76] Patricia P. Griffiths and Bradford W. Wade. An Authorization Mechanism for a Relational Database System. *ACM Trans. Database Syst.*, 1(3):242–255, September 1976. 3, 22

[HFK+13] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Arthur R. Friedman, Alan J. Lang, Margaret M. Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, Vincent C. Hu, David Ferraiolo, Rick Kuhn, Arthur R. Friedman, Alan J. Lang, Margaret M.

Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, and Scarfone Cybersecurity. Guide to attribute based access control (abac) definition and considerations (draft), 2013. 2

[HJPPW01] Åsa Hagström, Sushil Jajodia, Francesco Parisi-Presicce, and Duminda Wijesekera. Revocations-A Classification. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, CSFW '01, pages 44–, Washington, DC, USA, 2001. IEEE Computer Society. 3, 21, 22, 24, 26, 60, 155

[HM85] Joseph Y. Halpern and Yoram Moses. Towards a Theory of Knowledge and Ignorance: Preliminary Report. In Krzysztof R. Apt, editor, *Logics and Models of Concurrent Systems*, volume 13 of *NATO ASI Series*, pages 459–476. Springer Berlin Heidelberg, 1985. 70

[Hor93] John F. Horty. Deontic logic as founded on nonmonotonic logic. *Ann. Math. Artif. Intell.*, 9(1-2):69–91, 1993. 142, 146

[Hor94] John F. Horty. Moral dilemmas and nonmonotonic logic. *Journal of Philosophical Logic*, 23(1):35–65, 1994. 143

[IHC96] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. Lua – An Extensible Extension Language. *Software: Practice and Experience*, 26(6):635–652, 1996. 15

[JJ99] Bharat Jayaraman and Devashis Jana. Set constructors, finite sets, and logical semantics. *The Journal of Logic Programming*, pages 55–77, 1999. 43

[Kle38] S. C. Kleene. On Notation for Ordinal Numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938. 14

[Kon88] Kurt Konolige. On the relation between default and autoepistemic logic. *Artif. Intell.*, 35(3):343–382, 1988. 70

[Lev90] Hector J. Levesque. All I Know: A Study in Autoepistemic Logic. *Artif. Intell.*, 42(2-3):263–309, 1990. 64

[Moo85a] Robert C. Moore. A Formal Theory of Knowledge and Action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Springer-Verlag, 1985. 7

[Moo85b] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artif. Intell.*, 25(1):75–94, 1985. 64, 65, 68

[MP11] Matteo Maffei and Kim Pecina. Privacy-aware Proof-carrying Authorization. In *Proceedings of the ACM SIGPLAN 6th Workshop on Programming Languages and Analysis for Security*, PLAS '11, pages 7:1–7:6, New York, NY, USA, 2011. ACM. 2

[MvdT00] D. Makinson and L. van der Torre. Input-output logics. *Journal of Philosophical Logic*, 29(4):383–408, 2000. 116, 147, 148, 149, 150

[MvdT03] D. Makinson and L. van der Torre. Permissions from an input-output perspective. *Journal of Philosophical Logic*, 32(4):391–416, 2003. 125

[Nie91] Ilkka Niemelä. Constructive Tightly Grounded Autoepistemic Reasoning. In John Mylopoulos and Raymond Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*, pages 399–405. Morgan Kaufmann, 1991. 64

[NvP01] Sara Negri and Jan von Plato. *Structural proof theory*. Cambridge University Press, 2001. 113

[NvP11] Sara Negri and Jan von Plato. *Proof Analysis - A Contribution to Hilbert's Last Problem*. Cambridge University Press, 2011. 113

[PDJD16] Van Hertum Pieter, Ingmar Dasseville, Gerda Janssens, and Marc Denecker. *Proceedings, PADL 2016.*, chapter The KB Paradigm and Its Application to Interactive Configuration, pages 13–29. Cham, 2016. 7

[PM12] Henry Prakken and Sanjay Modgil. Clarifying some misconceptions on the ASPIC$^+$ framework. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Computational Models of Argument. Proceedings of COMMA 2012*, pages 442–453, 2012. 128

[Pri58] A. N. Prior. Escapism: The logical basis of ethics. In A. I. Melden, editor, *Journal of Symbolic Logic*, pages 610–611. University of Washington Press, 1958. 121

[PS96] Henry Prakken and Marek Sergot. Contrary-to-duty obligations. *Studia Logica: An International Journal for Symbolic Logic*, 57(1):91–115, 1996. 120, 152

[PvdT14a] X. Parent and L. van der Torre. "Sing and Dance!". In F. Cariani, D. Grossi, J. Meheus, and X. Parent, editors, *Deontic Logic and Normative Systems, DEON 2014*, volume 8554 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 2014. 116, 118, 119, 129, 133, 146, 147, 148, 149, 151, 153

[PvdT14b] Xavier Parent and Leon van der Torre. Aggregative deontic detachment for normative reasoning. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, 2014. 117

[Rau06] Wolfgang Rautenberg. *A Concise Introduction to Mathematical Logic*. Springer, 2006. 49

[Ros44] Alf Ross. Imperatives and logic. *Philosophy of Science*, 11(1):30–46, 1944. 121

[RV02] Chun Ruan and Vijay Varadharajan. Resolving Conflicts in Authorization Delegations. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 271–285. Springer, 2002. 25, 27

[Sar48] Jean-Paul Sartre. *Existentialism and Humanism*. Haskell House, 1948. 142

[SS94] R.S. Sandhu and P. Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, Sept 1994. 92

[Sto08] A. Stolpe. *Norms and Norm-System Dynamics*. PhD thesis, Department of Philosophy, University of Bergen, Norway, 2008. 151

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2):285–309, 1955. 18

[vdT97a] L. van der Torre. *Reasoning about obligations*. PhD thesis, Erasmus University Rotterdam, 1997. 121

[vdT97b] L.W.N. van der Torre. *Reasoning about obligations: defeasibility in preference-based deontic logic*. Tinbergen Institute research series. Thesis, 1997. 146

[vdT10] Leendert van der Torre. Deontic redundancy. In Guido Governatori and Giovanni Sartor, editors, *Deontic Logic in Computer Science: Proceedings of DEON 2010*, pages 11–32, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 147

[vdTT00] L. van der Torre and Y.H. Tan. Two-phase deontic logic. *Logique et Analyse*, 43:411–456, 2000. 146

[VF73] Bas C. Van Fraassen. Values and the heart's command. *Journal of Philosophy*, 70(1):5–19, 1973. 143

[VHCBD16] Pieter Van Hertum, Marcos Cramer, Bart Bogaerts, and Marc Denecker. Distributed Autoepistemic Logic and its Application to Access Control. In *Forthcomming. Proceedings of IJCAI 2016*, 2016. 7, 62, 64, 73, 77, 83, 84, 113

[vW51] G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951. 140, 141

[VW71] Georg Henrik Von Wright. *A New System of Deontic Logic**, pages 105–120. Springer Netherlands, Dordrecht, 1971. 140

[XvdT14] S. Xin and L. van der Torre. Combining constitutive and regulative norms in input/output logic. In F. Cariani, D. Grossi, J. Meheus, and X. Parent, editors, *Deontic Logic and Normative Systems, DEON 2014*, volume 8554 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2014. 125

Publications

[AAE+13] Diego Agustín Ambrossio, Alessio Antonini, Yehia Elrakaiby, Dov Gabbay, and Marc van Zee. Argument revival in annotated argumentation networks. In *Second workshop on Argumentation in Artificial Intelligence and Philosophy: computational and philosophical perspectives (ARGAIP-13)*, December 2013.

[APv16] Diego Agustín Ambrossio, Xavier Parent, and Leon van der Torre. A representation theorem for abstract cumulative aggregation. Technical report, 2016.

[AXL16] Diego Agustín Ambrossio, Parent Xavier, and van der Torre Leon. Cumulative aggregation. In Olivier Roy, Allard Tamminga, and Malte Willer, editors, *Deontic Logic and Normative Systems 13th International Conference*, pages 1–15. College Publications, 2016.

[CAH15] Marcos Cramer, Diego Agustín Ambrossio, and Pieter Van Hertum. A logic of trust for reasoning about delegation and revocation. In Edgar R. Weippl, Florian Kerschbaum, and Adam J. Lee, editors, *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, Vienna, Austria, June 1-3, 2015*, pages 173–184. ACM, 2015.

[SA15a] Xin Sun and Diego Agustín Ambrossio. Computational complexity of input/output logic. In Antonis Bikakis and Xianghan Zheng, editors, *Multi-disciplinary Trends in Artificial Intelligence - 9th International Workshop, MIWAI 2015, Fuzhou, China, November 13-15, 2015, Proceedings*, volume 9426 of *Lecture Notes in Computer Science*, pages 72–79. Springer, 2015.

[SA15b] Xin Sun and Diego Agustín Ambrossio. On the complexity of input/output logic. In Wiebe van der Hoek, Wesley H. Holliday, and Wen-Fang Wang, editors, *Logic, Rationality, and Interaction - 5th International Workshop, LORI 2015 Taipei, Taiwan, October 28-31, 2015, Proceedings*, volume 9394 of *Lecture Notes in Computer Science*, pages 429–434. Springer, 2015.