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

Diego Agustin Ambrossio
University of Luxembourg
2, avenue de l'Université
L-4365 Esch-sur-Alzette
Luxembourg
diego.ambrossio@uni.lu

Marcos Cramer
University of Luxembourg
2, avenue de l'Université
L-4365 Esch-sur-Alzette
Luxembourg
marcos.cramer@uni.lu

***Abstract*—Distributed Autoepistemic Logic with Inductive Definitions (dAEL(ID)) is a recently proposed non-monotonic logic for says-based access control. We define a query-driven decision procedure for dAEL(ID) that is implemented in the knowledge-base system IDP. The decision procedure is designed in such a way that it allows one to determine access rights while avoiding redundant information flow between principals in order to enhance security and reduce privacy concerns. Given that the decision procedure has in the worst case an exponential runtime, it is to be regarded as a proof of concept that increases our understanding of dAEL(ID), rather than being deployed for an access control system.**

***Index Terms*—access control, says-based logic, decision procedure, non-monotonic logic, autoepistemic logic, well-founded semantics, inductive definitions, IDP**

I. Introduction

Multiple logics have been proposed for distributed access control [1], [2], [3], [4], [5], most of which use a modality $k\,\mathit{says}$ indexed by a principal (i.e. user or process) k. These says-based access control logics are designed for systems in which different principals can issue statements that become part of the access control policy. $k\,\mathit{says}\,\varphi$ is usually rendered as "k supports φ", which can be interpreted to mean that 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.

Van Hertum et al. [6] have recently proposed a multi-agent variant of autoepistemic logic, called *Distributed Autoepistemic Logic with Inductive Definitions* (dAEL(ID)), to be used as a says-based access control logic. Autoepistemic logic is a non-monotonic logic originally designed for reasoning about knowledge bases and motivated by the principle that an agent's knowledge base completely characterizes what the agent knows [7]. By applying the semantic principles of autoepistemic logic to characterize the says-modality, dAEL(ID) allows us to derive a statement of the form $\neg k\,\mathit{says}\,\varphi$ on the basis of the observation that k has not issued statements implying φ. As explained in Section II-C, supporting reasoning about such negated says-statements allows dAEL(ID) to model access denials straightforwardly.

Van Hertum et al. have extended multiple semantics of autoepistemic logic to dAEL(ID), but have argued that the well-founded semantics is to be prefered in the application of dAEL(ID) to access control. In this paper we therefore restrict ourselves to the well-founded semantics of dAEL(ID).

When applying dAEL(ID) to access control, the access control policy consists of a separate set of dAEL(ID) formulas for each principal in the system, where the set of formulas of each principal consists of the statements issued by that principal. A principal k has access right to a resource r if and only if the owner j of that resource supports the formula $\mathit{access}(k,r)$, i.e. iff the dAEL(ID) formula $j\,\mathit{says}\,\mathit{access}(k,r)$ is true in the well-founded model of the access control policy.

We define a query-driven decision procedure for dAEL(ID), which – under the assumption of a finite domain – allows one to determine the truth value of a formula in the well-founded model of a dAEL(ID) access control policy, i.e. to determine access rights. This decision procedure is designed in such a way that it avoids redundant information flow between principals, which ensures that the need-to-know principle of computer security [8] is not violated, and which additionally reduces privacy concerns. This decision procedure is implemented with the help of the IDP system [9], a knowledge base system for the language of first-order logic with inductive definitions.

The decision procedure that we define has in the worst case an exponential runtime. This means that it is not practicable to build an access control system that implements this decision procedure without including heuristics to optimize response time and a principled approach for dealing with situations when access cannot be determined within a reasonable amount of time (see Section VII of Cramer et al. [10] for an example of such an approach in a somewhat different access-control setting). For this reason, we regard the contribution of this paper to be mainly conceptual: The defined decision procedure is a proof of concept that increases our understanding of dAEL(ID) by providing an algorithmic characterization of the well-founded semantics of dAEL(ID). This algorithmic characterization complements in a conceptually fruitful way the semantic definition from Van Hertum et al. [6] which is based on a fixpoint construction on abstract structures.

The rest of the paper is organized as follows. In Section II, we define dAEL(ID) and motivate its application to access control. In Section III, we introduce the IDP system and its language FO(ID). In Section IV, we present a query mechanism for determining access rights while avoiding redundant information flow between principals. Section V discusses related work. Section VI concludes the paper and presents possible future work.

II. Distributed Autoepistemic Logic with Inductive Definitions

Van Hertum et al. [6] have used two notational variants of dAEL(ID): In the first one, the modality of the logic is written as $K_A\varphi$, following the standard notation in autoepistemic logic. In the second one, it is written as $A\,says\,\varphi$, following the standard notation in access control logic. In this paper, we only use the notation $A\,says\,\varphi$.

A. dAEL(ID) Syntax

We assume that a set $\mathcal{A}$ of principals and a first-order vocabulary Σ consisting of function and predicate symbol with fixed arity is fixed throughout this paper. As usual, 0-ary function symbols play the role of constants, and 0-ary predicate symbols play the role of propositional variables. Terms are built from function symbols and variables in the usual manner.

Definition 1. *dAEL(ID) formulas are defined by the following EBNF rule, where P denotes a predicate symbol, t a term and x a 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. We follow the standard conventions for dropping brackets when this does not cause unclarity.

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

Definition 2. *A* $says$-atom *or* modal atom *is a formula of the form $t\,says\,\varphi$. A* $says$-literal *is a $says$-atom $t\,says\,\varphi$ or its negation $\neg t\,says\,\varphi$.*

As motivated in Section II-C below, dAEL(ID) contains a construct for inductive definitions:

Definition 3. *We define a dAEL(ID)* inductive definition Δ *to be a finite 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 dAEL(ID) formula. $P(\overline{x})$ is called the* head *and $\varphi(\overline{y})$ the* body *of the rule $\forall\overline{x} : P(\overline{x}) \leftarrow \varphi(\overline{y})$.*

Definition 4. *A dAEL(ID)* theory T *is a set that consists of dAEL(ID) formulas and dAEL(ID) inductive definitions.*

In a distributed setting, different principals can issue statements that become part of the access control policy. A dAEL(ID) theory as defined above only represent the set of statements of the access control policy issued by a single principal. In order to represent the full access control policy, we use the notion of a *distributed theory*:

Definition 5. *A* distributed theory $\mathbb{T}$ *is an indexed family $(\mathbb{T}_A)_{A\in\mathcal{A}}$, where each $\mathbb{T}_A$ is a dAEL(ID) theory.*

B. Semantics

Van Hertum et al. [6] have defined various semantics for dAEL(ID) using Approximation Fixpoint Theory [11], but have argued for the use of the well-founded semantics in the application of dAEL(ID) to access control. In this paper, we define a decision procedure for dAEL(ID) with respect to the well-founded semantics, so we only define this semantics. The definition of the semantics involves a lot of technical machinery, but for a reader familiar with autoepistemic logic, it is enough to know that the well-founded semantics of dAEL(ID) is an extension of the well-founded semantics of autoepistemic logic [12] to the multi-agent case under the assumption of mutual positive and negative introspection between the agents. We motivate this mutual introspection below in Section II-C. Note that the well-founded semantics of dAEL(ID) is defined over a fixed domain D, which can be either finite or infinite (but for the decision procedure in Section IV, D is assumed to be finite).

For defining the well-founded semantics of dAEL(ID), we use the methodology of *Approximation Fixpoint Theory* that Denecker et al. [11] used to define the well-founded semantics of autoepistemic logic. This methodology is based on the idea of approximating the knowledge of an agent using a three-valued valuation, in which formulas, inductive definitions and theories may not only be true or false but also undefined. The logical connectives combine these three truth values based on Kleene's truth tables [13].

We use truth values $\mathbf{t}$ for truth, $\mathbf{f}$ for falsity, and $\mathbf{u}$ 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 $\mathbf{u}^{-1} = \mathbf{u}$.

A *structure* is defined as usual in first-order logic:

Definition 6. *A* structure I *consists of a set D, called the* domain *of I, an assignment that maps every n-ary predicate symbol of Σ to a subset of D^n and an assignment that maps every n-ary function symbol of Σ to a function $D^n \to D$.*

A structure formally represents a potential state of affairs of the world. The interpretation of a term in a structure is defined as usual.

We assume a domain D, shared by all structures, to be fixed throughout the paper. Furthermore, we assume D to contain the set $\mathcal{A}$ of principals.

The semantics of dAEL(ID) is based on the truth assignment of S5 modal logic, extended to the multi-agent case in such a way that mutual positive and negative introspection between agents is satisfied. While S5 modal logic is often used for formalizing the knowledge modality, we make indirect usage of it for formalizing the $says$-modality. But for convenience, we will sometimes use knowledge terminology when informally explaining the formal definitions needed for defining dAEL(ID) semantics.

The following notion is used to model a single agent's knowledge:

Definition 7. *A* possible world structure *Q is a set of structures.*

Note that a possible world structure can be seen as a Kripke structure with the total accessibility relation. It contains all structures that are consistent with an agent's knowledge.

Possible world structures are ordered with respect to the amount of knowledge they contain. In this sense, possible world structures that contain less structures possess more knowledge:

Definition 8. *Given two possible world structures Q_1 and Q_2, we define $Q_1 \leq_K Q_2$ to hold if and only if $Q_1 \supseteq Q_2$.*

In order to model the interaction of the knowledge of multiple agents, we extend the notion of a possible world structure to the multi-agent case as follows:

Definition 9. *A* distributed possible world structure *(*DPWS*) $\mathcal{Q} = (\mathcal{Q}_A)_{A\in\mathcal{A}}$ is a family consisting of a possible world structure $\mathcal{Q}_A$ for each principal $A \in \mathcal{A}$.*

The knowledge order on possible world structures can be extended pointwise to DPWS's. One DPWS contains more knowledge than another if each principal has more knowledge:

Definition 10. *Given two DPWS's $\mathcal{Q}^1$ and $\mathcal{Q}^2$, we define $\mathcal{Q}^1 \leq_K \mathcal{Q}^2$ iff $\mathcal{Q}^1_A \leq_K \mathcal{Q}^2_A$ for each $A \in \mathcal{A}$.*

Definition 11. *We inductively define a two-valued valuation of dAEL(ID) formulas with respect to a DPWS $\mathcal{Q}$ and a structure I as follows:*

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

Inductive definitions generally define only some of the predicates of a language, while the remaining predicates of the language function as parameters:

Definition 12. *Let $\Delta = \{P_1(\bar{t}_1) \leftarrow \varphi_1, \ldots, P_n(\bar{t}_n) \leftarrow \varphi_n\}$ be an inductive definition. Then $Def(\Delta)$ 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 $Def(\Delta)$ is denoted $Par(\Delta)$ and is called the set of* parameters *of Δ.*

In order to approximate the agents' knowledge in a three-valued setting, we use *distributed belief pairs* that consist of a conservative bound $\mathcal{B}^c$ and a liberal bound $\mathcal{B}^l$ of each agent's knowledge, i.e. it specifies what each agent knows for certain and what each agent possibly knows:

Definition 13. *A* distributed belief pair *$\mathcal{B}$ is a pair $(\mathcal{B}^c, \mathcal{B}^l)$ of two DPWS's $\mathcal{B}^c$ and $\mathcal{B}^l$ such that $\mathcal{B}^c \leq_K \mathcal{B}^l$.*

The knowledge order $\leq_K$ on DPWS's induces a precision order $\leq_p$ on distributed belief pairs:

Definition 14. *Given two distributed belief pairs $\mathcal{B}_1$ and $\mathcal{B}_2$, we define $\mathcal{B}_1 \leq_p \mathcal{B}_2$ to hold iff $\mathcal{B}_1^c \leq \mathcal{B}_2^c$ and $\mathcal{B}_2^l \leq \mathcal{B}_1^l$.*

Intuitively, $\mathcal{B}_1 \leq_p \mathcal{B}_2$ means that $\mathcal{B}_2$ characterizes the knowledge of the principals more precisely than $\mathcal{B}_1$.

Definition 15. *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:*

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

As explained in Section II-C below, inductive definitions in dAEL(ID) are interpreted according to the well-founded semantics for inductive definitions, as defined for example in [14]. The well-founded model of an inductive definition Δ is always defined relative to a context $\mathcal{O}$, which is an interpretation of the predicate symbols in $Par(\Delta)$. 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. Since the definition of the well-founded model in [14] is only defined for inductive definition over a first-order language without any modality, we need to say something about how to interpret the $says$-modality in the body. Just like formulas, 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 **t** or **f** according to their interpretation in $\mathcal{Q}$.

This motivates the following definition of a three-valued valuation of dAEL(ID) inductive definitions with respect to a DPWS $\mathcal{Q}$ and a structure I:

Definition 16. *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:*

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

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

To understand this three-valued valuation of dAEL(ID) inductive definitions informally, remark that in a partial context ($\mathcal{B}$ is three-valued), we cannot yet evaluate the exact value of the defined predicates in the definition. We can, however, using a three-valued valuation of the definition, obtain an approximation $wfm_{\Delta^{\mathcal{B}}}(I|_{Par(\Delta)})$ of their value. We return **t** if this approximation is actually two-valued and equal to I, **u** if I is still consistent with (but not equal to) this approximation and **f** otherwise.

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 17. *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:*

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

Using this three-valued valuation of dAEL(ID) theories, we can define an operator $\mathcal{D}^*_{\mathbb{T}}$ on distributed belief pairs:

Definition 18. $\mathcal{D}^*_{\mathbb{T}}(\mathcal{B}) := (\mathcal{D}^c_{\mathbb{T}}(\mathcal{B}), \mathcal{D}^l_{\mathbb{T}}(\mathcal{B}))$, *where*

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

In order to formally define the well-founded model, we first need to define the *stable operator* $S_{\mathbb{T}}$ that maps a DPWS to a DPWS:

Definition 19. $S_{\mathbb{T}}(\mathcal{Q})$ *is defined to be the least fixpoint of the operator O that maps a DPWS $\mathcal{Q}'$ to the DPWS $O(\mathcal{Q}) := \mathcal{D}^*_{\mathbb{T}}(\mathcal{Q}', \mathcal{Q})_1$, i.e. to the first element of the distributed belief pair $\mathcal{D}^*_{\mathbb{T}}(\mathcal{Q}', \mathcal{Q})$.*

Now we are ready to define the well-founded model of a distributed theory, the central notion of dAEL(ID) semantics:

Definition 20. *Let $\mathbb{T}$ be a distributed theory. The well-founded model of $\mathbb{T}$, denoted wfm($\mathbb{T}$), is the least precise (i.e. $\leq_p$-minimal) distributed belief pair $\mathcal{B}$ such that $S_{\mathbb{T}}(\mathcal{B}^c) = \mathcal{B}^l$ and $S_{\mathbb{T}}(\mathcal{B}^l) = \mathcal{B}^c$.*

We say that a distributed theory logically implies a formula φ iff $\varphi^{wfm(\mathbb{T}),I} = \mathbf{t}$ for every structure I.

Note that for a formula φ of the form $k\, says\, \psi$ or $\neg k\, says\, \psi$, the value of $\varphi^{wfm(\mathbb{T}),I}$ does not depend on I. We therefore sometimes write $\varphi^{wfm(\mathbb{T})}$ instead of $\varphi^{wfm(\mathbb{T}),I}$ for such φ.

C. Motivation for dAEL(ID)

Van Hertum et al. [6] have motivated the applicability of dAEL(ID) as an access control logic by discussing possible use cases, i.e. by illustrating how dAEL(ID) can be applied in certain access control scenarios. In this section we add to this motivation by use cases a more principled motivation that clarifies the advantages of dAEL(ID) over other $says$-based access control logics.

An *access control policy* is a set of norms defining which principal is to be granted access to which resource under which circumstances. Specialized logics called *access control logics* were developed for representing policies and access requests and reasoning about them. A general principle adopted by most logic-based approaches to access control is that access is granted iff it is logically entailed by the policy.

There is a large variety of access control logics, but most of them use a modality $k\, says$ indexed by a principal k [5]. $says$-based access control logics are designed for systems in which different principals can issue statements that become part of the access control policy. $k\, says\, \varphi$ is usually explained informally to mean that k supports φ [3], [4], [5]. This means that k has issued statements that – together with additional information present in the system – imply φ. 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.

Many state-of-the-art $says$-based access control logics, e.g. Garg's BL [4], 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 him 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 [7], [15], [16], [17]. 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. [6] 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 [9]. Denecker [18] showed that in classical logics, adding definitions leads to a strictly more expressive language.

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.

Most of the semantics that have been proposed for logic programs can be adapted to inductive definitions. Denecker and Venneckens [14] 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.

III. FO(ID) AND THE IDP-SYSTEM

The decision procedure defined in the next section is based on the IDP system, so we briefly describe this system and its language FO(ID).

A. *Why* IDP*?*

IDP [9] is a Knowledge Base System which combines a declarative specification (*knowledge base*), written in an extension of first-order logic, with an imperative management of the specification via the Lua [19] scripting language. The extension of first-order logic supported by IDP allows for inductive definitions. As explained and motivated in Section II-C, dAEL(ID) also supports inductive definitions. This makes the usage of IDP as a basis for the decision procedure a natural choice.

IDP supports multiple *inferences* that can be used to perform a range of reasoning tasks on a given specification. We make use of two of IDP's inferences, defined in Section III-B below, in order to perform the meta-reasoning about a principal's dAEL(ID) theory that is necessary to determine which queries to other principals are really necessary in order to resolve a query asked to the principal.

B. FO(ID) *and some* IDP *inferences*

The specification language supported by IDP is an extension of first-order logic (FO) with types, inductive definitions, aggregates, arithmetic and partial functions, denoted FO(T,ID,Agg,Arit,PF) [20]. We only make use of the subset of FO(T,ID,Agg,Arit,PF) called FO(ID), which extends FO only with inductive definitions. The formal definition of FO(ID) syntax is the standard definition of FO syntax extended by the following definition of inductive definitions: 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 an FO formula. Just as in dAEL(ID), inductive definitions are given the well-founded semantics of inductive definitions [14]. An FO(ID) theory is a set of inductive definitions and FO formulas.

The IDP inferences for FO(ID) that we make use of are defined for *finite partial structures*. Before we define formally what a partial structure is, we define the concept of a *partial set*, a generalization of a set in a three-valued context:

Definition 21. *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.

Given a vocabulary Σ, a partial structure gives an interpretation to the elements of Σ:

Definition 22. *A* partial structure *over our fixed vocabulary Σ is a tuple $(D, \mathcal{I})$, where the domain D is a set, 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 partial set on the domain D^n; for a function symbol f of arity n, $f^{\mathcal{I}}$ is a function from D^n to D.*

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 $S = (D, \mathcal{I})$ *finite* iff its domain D is finite. We call a partial structure *total* iff $P^{\mathcal{I}}$ is total for all $P \in \Sigma$.

The interpretation of terms $t^{\mathcal{I}}$ and the satisfaction relation $\models$ for total structures $S \models \varphi$ are defined as usual.

A precision order can be defined on partial structures:

Definition 23. *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, $f^{\mathcal{I}'} = f^{\mathcal{I}}$, 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 $P^{\mathcal{I}'}(\bar{d}) = P^{\mathcal{I}}(\bar{d})$.*

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 24. *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 25. *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 theory $\mathcal{T}$, then $min\_incons\_set(\mathcal{T}, S)$ 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.

IV. Decision Procedure

In this section, we define a query-driven decision procedure for dAEL(ID), which allows to determine access rights while avoiding redundant 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.[1] 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. Motivation for avoiding redundant information flow

One reason to avoid redundant information flow is to reduce communication overhead. The rest of this section considers an additional motivation for avoiding redundant information flow.

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

$$T_A = \left\{ \begin{array}{l} r \wedge B\ says\ s \Rightarrow p \\ r \end{array} \right\}$$

$$T_B = \left\{ \begin{array}{l} s \\ \neg s \wedge A\ says\ p \Rightarrow p \end{array} \right\}$$

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 [8], which states that a principal should only be given those accesses and be provided with that non-public information 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 avoids redundant 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 Section IV-E, this ideal avoidance of redundant communication is computationally very expensive, so in a practically applicable system, a trade-off between the security and privacy motivation for avoiding redundant communication on the one hand and computational cost on the other hand would need to be found. Nevertheless, we consider our ideal avoidance of redundant 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 $\mathbb{T}$ 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 handling of the loops that may occur.

B. Query Minimization Procedure

1) 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

[1] 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.

propositional variables of the form $p^{+}_{A\_\mathsf{says}\_\varphi}$, $p^{-}_{A\_\mathsf{says}\_\varphi}$ and $w_{A\_\mathsf{says}\_\varphi}$ 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 $p^{+}_{A\_\mathsf{says}\_\varphi}$, $p^{-}_{A\_\mathsf{says}\_\varphi}$ and $w_{A\_\mathsf{says}\_\varphi}$ 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 $p^{+}_{A\_\mathsf{says}\_\varphi}$ and $p^{-}_{A\_\mathsf{says}\_\varphi}$ 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 **t** (*true*), **f** (*false*) and **u** (*undefined*) induced by $\mathbf{u} <_p \mathbf{t}$ and $\mathbf{u} <_p \mathbf{f}$, the propositional variable $p^{+}_{A\_\mathsf{says}\_\varphi}$ represents the upper bound for the truth value of $A\,says\,\varphi$ and $p^{-}_{A\_\mathsf{says}\_\varphi}$ the lower bound. For this reason, we replace every positive occurrence of $A\,says\,\varphi$ by $p^{+}_{A\_\mathsf{says}\_\varphi}$, and every negative occurrence by $p^{-}_{A\_\mathsf{says}\_\varphi}$. 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 $w_{A\_\mathsf{says}\_\varphi}$ and add two implications to the theories that express the equivalence between $w_{A\_\mathsf{says}\_\psi}$ and $A\,says\,\varphi$.

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

Definition 26. *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 $w_{A\_\mathsf{says}\_\varphi}$*
- *Add to $t(T)$ the two formulae $w_{A\_\mathsf{says}\_\varphi} \Rightarrow A\,says\,\varphi$ and $A\,says\,\varphi \Rightarrow w_{A\_\mathsf{says}\_\varphi}$.*

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

Definition 27. *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 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 28. *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 29. *Let T be a dAEL(ID) theory. $\tau(T)$ is constructed from $t(T)$ by performing the following replacements for every says-atom $A\,says\,\varphi$ occurring in $t(T)$ that is not a subformula of another says-atom:*

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

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

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

$$T_B = \left\{ \begin{array}{l} p \\ C\ \ says\ \ z \Rightarrow z \\ C\ \ says\ \ r \Rightarrow r \end{array} \right\}$$

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

We translate these theories as follows:

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

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

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

2) 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 $\mathbb{L} = \{\}$, the query necessarily fails, whereas if $\mathbb{L} = \{\{\}\}$ (contains the empty set), 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 $p^{-}_{A\_\mathsf{says}\_\varphi}$ and $p^{+}_{A\_\mathsf{says}\_\varphi}$. Taking into account that $p^{-}_{A\_\mathsf{says}\_\varphi}$ and $p^{+}_{A\_\mathsf{says}\_\varphi}$ 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 30. *For a partial structure $S = (D, \mathcal{I})$, we define L^S to be*

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

We say that a $says$-atom $A\, says\, \varphi$ *occurs directly* in a dAEL(ID) theory, if some occurrence of $A\, says\, \varphi$ in $\mathbb{T}$ 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 $p^+_{A\_\mathsf{says}\_\varphi}$ and $S \models p^-_{A\_\mathsf{says}\_\varphi}$, and for which this information is coherent in the sense that the truth values assigned to $p^+_{A\_\mathsf{says}\_\varphi}$ and $S \models p^-_{A\_\mathsf{says}\_\varphi}$ 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 31. *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). Please note that lines 4 and 5 are implemented using the IDP inferences `sat` and `unsatstructure` that we defined in Section III-B.

Algorithm 1 Query Minimization Procedure

```
Input: theory $T$, dAEL(ID) query $\alpha$
Output: set $\mathbb{L}$ of sets of modal atoms
1: $\mathbb{L} := \emptyset$
2: $\mathcal{T} := \tau(T \cup \{\neg\alpha\})$
3: for each $S \in \mathbb{S}_T$ do
4:     if $S$ is not a partial model of $\mathcal{T}$ then
5:         pick a partial structure $S_{min}$ from $min\_incons\_set(\mathcal{T}, S)$
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 theory T and the negation of the query α into an augmented FO(ID) theory $\mathcal{T}$. 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 $\mathcal{T}$; note that the information in such a structure S together with the information in T entails the query α. Furthermore, note that for such a structure S, $min\_incons\_set(\mathcal{T}, S)$ is non-empty. So next (line 5), we pick a structure S_{min} from $min\_incons\_set(\mathcal{T}, S)$; by definition S_{min} is a minimal structure such that $S_{min} \leq_p S$ and S_{min} is not a partial model of $\mathcal{T}$; this means that S_{min} contains a minimal amount of information from S that together with the information in 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.[2] 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 1 to illustrate the query minimization procedure.

Example 2. *We apply the Query Minimization Procedure to the theory T_A and the query z. First we translate the theory T_A and the negation of the query into $\mathcal{T} = \tau(T_A \cup \{\neg z\})$, as shown in Example 1 with the addition of the formula $\neg z$, since the query does not contain any says-atoms. 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 $\mathcal{T}$, because $p^-_{B\_says\_p}$ is inconsistent with $p^-_{B\_says\_r} \vee \neg p^+_{B\_says\_r} \Rightarrow z$ and z. Now $min\_incons\_set(\mathcal{T}, S)$ 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 $\{\{B\, says\, r\}, \{B\, says\, p, B\, says\, z\}, \{\neg B\, says\, r\}\}$.

C. 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 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 32. *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*

[2] Lemma 1 in Appendix B makes this claim more precise.

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 $k\, says\, \varphi$ or $\neg k\, says\, \varphi$.*
- *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 $k\, says\, \alpha$ or the literal $\neg k\, says\, \alpha$ 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 2, which calls the main recursive procedure defined under Algorithm 3.

Algorithm 2 Communication Procedure Initialization

```
Input: distributed theory $\mathbb{T}$, principal $A$, dAEL(ID) formula $\alpha$
Output: truth-value $V \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$
$G$ := the labelled graph consisting only of a single vertex $v$ labelled $\langle A : \alpha \rangle$ and no edges
$G$ := Communication_Procedure($\mathbb{T}$,$G$,$v$)
$V$ := the label on the query vertex $\langle A : \alpha \rangle$ in $G$
return $V$
```

Informally, the Communication Procedure can be explained as follows: The Query Minimization Procedure is called for $\mathbb{T}_A$ 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 $\mathbf{t}$ or $\mathbf{f}$ 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 $\mathbf{t}$ (lines 23-24). There is a dual procedure for labelling query vertices with $\mathbf{f}$ (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 $\mathbf{f}$ or $\mathbf{u}$, depending on whether the loop is over a negation (i.e. there is an $\mathbf{f}$-labelled edge in the path connecting the two vertices with the same label) or not (lines 16-20). $\mathbf{u}$-labels can also propagate towards the root of the graph (line 28).

We continue Example 1 to illustrate the Communication Procedure.

Example 3. *Given the distributed theory $\mathbb{T} = \{T_A, T_B, T_C\}$, we query the principal A about the truth value of z. We show the final graph in Figure 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 $\mathbb{T}$, G and v, chich means that we must call the Query Minimization Procedure, which returns the set $\mathbb{L} = \{\{B\, says\, p, B\, says\, z\}, \{B\, says\, r\}, \{\neg B\, says\, r\}\}$ as shown in Example 2. 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 $\{B\, says\, p, B\, says\, z\}$ is added to G with its corresponding edge. Now we consider each $says$-literal in the vertex.*

(i) For literal $B\, says\, p$ 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 $B\, says\, z$ 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 $\mathbb{L} = \{\{C\, says\, z\}\}$; 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 $\mathbb{L}'' = \{\{\neg B\, says\, z\}\}$; 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 neither $\mathbf{t}$ 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 $\{B\, says\, r\} \in \mathbb{L}$.

Algorithm 3 Communication Procedure

```
Input: distributed theory $\mathbb{T}$, query graph $G$, query vertex $v$ of $G$,
Output: updated query graph $G$
k := the principal mentioned in the label of $v$
$\varphi$ := the formula mentioned in the label of $v$
$\mathbb{L}$ := Query_Minimization_Procedure($\mathbb{T}_k$,$\varphi$)
while the input query vertex $v$ does not have a truth-value attached to it do
    for $L \in \mathbb{L}$ do
        add a new $says$-literal set vertex $L$ to $G$
        add to $G$ a new edge from vertex $v$ to vertex $L$
        for $l \in L$ do
            $k'$ := the principal such that $l$ is of the form $k'\, says\, \psi$ or $\neg k'\, says\, \psi$
            $\psi$ := the formula such that $l$ is of the form $k'\, says\, \psi$ or $\neg k'\, says\, \psi$
            add a query vertex $v'$ labelled by $\langle k' : \psi \rangle$ to $G$
            if $l$ is $k'\, says\, \psi$ then
                add to $G$ a new edge labelled t from vertex $L$ to vertex $\langle k' : \psi \rangle$
            if $l$ is $\neg k'\, says\, \psi$ then
                add to $G$ a new edge labelled f from vertex $L$ to vertex $\langle k' : \psi \rangle$
            if a query vertex $v''$ that is an ancestor of $v'$ is also labelled $\langle k' : \psi \rangle$ then
                if all labelled edges between $v''$ and $v'$ are labelled by t then
                    add f-label to $v'$
                else
                    add u-label to $v'$
            else
                Communication_Procedure($\mathbb{T}$,$G$,$v'$)
        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
            label $v$ with t
    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
        label $v$ with f
    else
        label $v$ with u
return  $G$
```

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 **f***, so the truth-value assignment for the vertex at which the loop is detected is* **f***. Again this truth-value is propagated to label the two query vertices above this vertex, as the labelled edges are labelled by* **t***. Since the truth-value* **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* **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 **f** *instead of* **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* **t***. This ends the main while loop and therefore the Communication Procedure. Finally, the Communication Procedure Initialization returns the output* **t**.

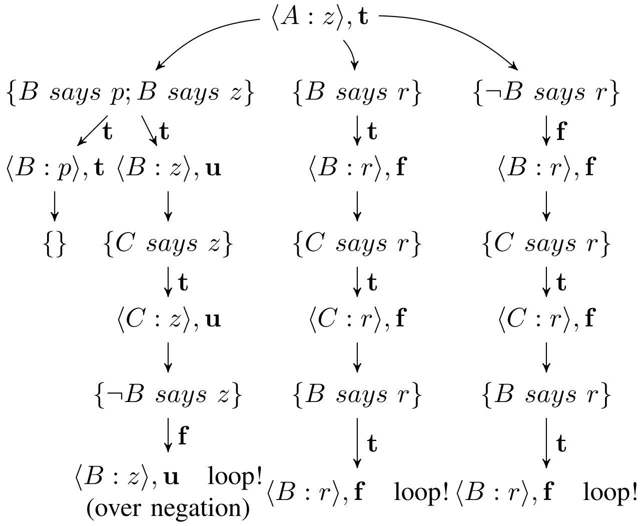

Fig. 1. Query Graph

D. Correctness of decision procedure

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

Theorem 1. *Let $\mathbb{T}$ be a distributed theory, let A be an agent, and let α be a dAEL(ID) formula. When A's theory $\mathbb{T}_A$ is queried about α, the decision procedure returns $(A\, says\, \alpha)^{wfm(\mathbb{T})}$, i.e. the truth value of $A\, says\, \alpha$ in the well-founded model of $\mathbb{T}$.*

The proof of Theorem 1 can be found in the Appendix.

E. Complexity of the decision procedure

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 has runtime quasilinear in the number of subqueries called.

To make the decision procedure practically applicable, heuristics would have to employed to reduce the runtime for determining an access right, and a principled approach for dealing with situations when access cannot be determined within a reasonable amount of time would be required (see Section VII of Cramer et al. [10] for an example of such an approach in a somewhat different access-control setting). 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.

V. 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 [6] 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 [21], [22] and Genovese [5] 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 [5] follows the methodology of Negri and von Plato [23], [24] 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. [12] 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 IV-A).

VI. Conclusion and Future Work

In this paper, 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 one to determine access rights while avoiding redundant information flow between principals in order to enhance security and reduce privacy concerns.

Given that our decision procedure has in the worst case an exponential runtime (see Section IV-E), 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. For this reason, the contribution of this paper is mainly of a conceptual nature: The defined decision procedure is a proof of concept that increases our understanding of dAEL(ID) by providing an algorithmic characterization of the well-founded semantics of dAEL(ID). This algorithmic characterization complements in a conceptually fruitful way the semantic definition from Van Hertum et al. [6] which is based on a fixpoint construction on abstract structures.

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 [25]. Hence it may be interesting to define justification semantics for dAEL(ID).

References

[1] M. Abadi, "Logic in Access Control," in *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science*, 2003, pp. 228–233.

[2] Y. Gurevich and I. Neeman, "DKAL: Distributed-knowledge authorization language," in *Proceedings of the 2008 21st IEEE Computer Security Foundations Symposium*. IEEE, 2008, pp. 149–162.

[3] M. Abadi, "Variations in Access Control Logic," in *9th International Conference on Deontic Logic in Computer Science*, 2008, pp. 96–109.

[4] D. Garg and F. Pfenning, "Stateful Authorization Logic – Proof Theory and a Case Study," *Journal of Computer Security*, vol. 20, no. 4, pp. 353–391, 2012.

[5] V. Genovese, "Modalities in Access Control: Logics, Proof-theory and Application," Ph.D. dissertation, 2012.

[6] P. Van Hertum, M. Cramer, B. Bogaerts, and M. Denecker, "Distributed Autoepistemic Logic and its Application to Access Control," in *Forthcomming. Proceedings of IJCAI 2016*, 2016.

[7] R. C. Moore, "Semantical considerations on nonmonotonic logic," *Artif. Intell.*, vol. 25, no. 1, pp. 75–94, 1985. [Online]. Available: http://dx.doi.org/10.1016/0004-3702(85)90042-6

[8] R. Sandhu and P. Samarati, "Access control: principle and practice," *Communications Magazine, IEEE*, vol. 32, no. 9, pp. 40–48, Sept 1994.

[9] B. De Cat, B. Bogaerts, M. Bruynooghe, and M. Denecker, "Predicate Logic as a Modelling Language: The IDP System," *CoRR*, vol. abs/1401.6312, 2014. [Online]. Available: http://arxiv.org/abs/1401.6312

[10] M. Cramer, P. V. Hertum, R. Lapauw, I. Dasseville, and M. Denecker, "Resilient Delegation Revocation with Precedence for Predecessors Is NP-Complete," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, June 2016, pp. 432–442.

[11] M. Denecker, V. Marek, and M. Truszczyński, "Fixpoint 3-valued semantics for autoepistemic logic," in *AAAI'98*. Madison, Wisconsin: MIT Press, July 26-30 1998, pp. 840–845. [Online]. Available: http://www.aaai.org/Papers/AAAI/1998/AAAI98-119.pdf

[12] ——, "Uniform semantic treatment of default and autoepistemic logics," *Artif. Intell.*, vol. 143, no. 1, pp. 79–122, 2003. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(02)00293-X

[13] S. C. Kleene, "On Notation for Ordinal Numbers," *The Journal of Symbolic Logic*, vol. 3, no. 4, pp. 150–155, 1938. [Online]. Available: http://www.jstor.org/stable/2267778

[14] M. Denecker and J. Vennekens, "The Well-Founded Semantics Is the Principle of Inductive Definition, Revisited," in *KR*, C. Baral, G. De Giacomo, and T. Eiter, Eds. AAAI Press, 2014, pp. 1–10. [Online]. Available: http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7957

[15] H. J. Levesque, "All I Know: A Study in Autoepistemic Logic," *Artif. Intell.*, vol. 42, no. 2-3, pp. 263–309, 1990. [Online]. Available: http://dx.doi.org/10.1016/0004-3702(90)90056-6

[16] I. Niemelä, "Constructive Tightly Grounded Autoepistemic Reasoning," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*, J. Mylopoulos and R. Reiter, Eds. Morgan Kaufmann, 1991, pp. 399–405.

[17] M. Denecker, V. Marek, and M. Truszczyński, "Reiter's Default Logic Is a Logic of Autoepistemic Reasoning And a Good One, Too," in *Nonmonotonic Reasoning – Essays Celebrating Its 30th Anniversary*, G. Brewka, V. Marek, and M. Truszczyński, Eds. College Publications, 2011, pp. 111–144. [Online]. Available: http://arxiv.org/abs/1108.3278

[18] M. Denecker, "Extending Classical Logic with Inductive Definitions," in *CL*, ser. LNCS, J. W. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, and P. J. Stuckey, Eds., vol. 1861. Springer, 2000, pp. 703–717.

[19] R. Ierusalimschy, L. Henrique de Figueiredo, and W. Celes, “Lua – An Extensible Extension Language,” *Software: Practice and Experience*, vol. 26, no. 6, pp. 635–652, 1996. [Online]. Available: http://dx.doi.org/10.1002/(SICI)1097-024X(199606)26:6⟨635::AID-SPE26⟩3.0.CO;2-P

[20] V. H. Pieter, I. Dasseville, G. Janssens, and M. Denecker, *Proceedings, PADL 2016.*, Cham, 2016, ch. The KB Paradigm and Its Application to Interactive Configuration, pp. 13–29.

[21] D. Garg and M. 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.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ch. A Modal Deconstruction of Access Control Logics, pp. 216–230. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78499-9_16

[22] D. Garg, “Principal-Centric Reasoning in Constructive Authorization Logic,” 2008.

[23] S. Negri and J. von Plato, *Structural proof theory.* Cambridge University Press, 2001.

[24] ——, *Proof Analysis - A Contribution to Hilbert’s Last Problem.* Cambridge University Press, 2011.

[25] M. Denecker, G. Brewka, and H. Strass, “A Formal Theory of Justifications,” in *Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, ser. Lecture Notes in Computer Science, F. Calimeri, G. Ianni, and M. Truszczynski, Eds., vol. 9345. Springer, 2015, pp. 250–264. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23264-5_22

Appendix

The following lemma states that the Query Minimization Procedure (Algorithm 1) really does what it is supposed to do:

Lemma 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*

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

Let $\bot$ denote the DPWS in which every agent’s possible world structure is the set of all structures over domain D and vocabulary Σ. Let $\top$ denote the DPWS in which every agent’s possible world structure is the empty set.

The well-founded model of $\mathbb{T}$ is the $\leq_p$-least fixpoint of $S_{\mathbb{T}}$. When the domain is finite, as we are assuming when applying the decision procedure, there is a natural number n such that $\textit{wfm}(\mathbb{T}) = (S_{\mathbb{T}})^n(\bot, \top)$. In other words, the well-founded model can be computed by a finite number of application of $S_{\mathbb{T}}$ to $(\bot, \top)$, until a fixpoint is reached.

The steps in the decision procedure defined in section IV do not directly correspond to the steps in the computation of the well-founded model by a finite number of application of $S_{\mathbb{T}}$ 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 IV, but whose steps correspond more directly to the iterative application of $S_{\mathbb{T}}$ to $(\bot, \top)$. We call this auxiliary decision procedure the *$S_{\mathbb{T}}$-based decision procedure*. So we prove Theorem 1 by proving two things:

- The decision procedure defined in section IV is equivalent to the $S_{\mathbb{T}}$-based decision procedure.
- When A’s theory $\mathbb{T}_A$ is queried about α, the $S_{\mathbb{T}}$-based decision procedure returns yes iff $(A \textit{ says } \alpha)^{\textit{wfm}(\mathbb{T})} = \mathbf{t}$.

In the definition of the $S_{\mathbb{T}}$-based decision procedure, we use a *query graph* as defined in section IV.

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

Definition 33. *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 $(k \textit{ says } \varphi)^{\mathcal{B}}$.*

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 $S_{\mathbb{T}}$-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 4.

Algorithm 4 Make labelling of query vertices good

```
Input: query graph $G$
Output: modified query graph $G$
while there is a **u**-labelled query vertex $k : \varphi$ in $G$ such that replacing $says$-atoms in $\varphi$ corresponding to **t**- or **f**-labelled query vertices by **t** and **f** respectively makes $\varphi$ a tautology do
    change the **u**-label in each such query vertex in $G$ by **t**
while there is a **f**-labelled query vertex $k : \varphi$ in $G$ such that replacing $says$-atoms in $\varphi$ corresponding to **u**-, **t**- or **f**-labelled query vertices by **t** or **f**, **t** and **f** respectively makes $\varphi$ a tautology do
    change the **f**-label in each such query vertex in $G$ by **u**
return $G$
```

In order to define the $S_{\mathbb{T}}$-based decision procedure, we furthermore need the following two definitions:

Definition 34. *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 35. *In a query graph, a says-literal set vertex L is defined to be* dissatisfied *if either for some **t**-labelled edge from L to a query vertex, the query vertex is labelled by **f**, or for some **f**-labelled edge from L to a query vertex, the query vertex is labelled **t**.*

The definition of the $S_{\mathbb{T}}$-based decision procedure is given by the pseudo-code under Algorithm 5.

Algorithm 5 $S_{\mathbb{T}}$-based decision procedure

```
Input: distributed theory $\mathbb{T}$, principal $A$, dAEL(ID) formula $\alpha$
Output: truth-value $V \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$
 1: $G$ := the empty graph
 2: add a new query vertex $A : \alpha$ to $G$
 3: query_stack := $\langle A : \alpha \rangle$
 4: while query_stack $\neq \langle\rangle$ do
 5:   $k : \varphi$ := first element of query_stack
 6:   $\mathbb{L}$ := Query_Minimization_Procedure($\mathbb{T}_k$,$\varphi$)
 7:   for $L \in \mathbb{L}$ do
 8:     if $G$ does not contain a $says$-literal set vertex $L$ then
 9:       add a new $says$-literal set vertex $L$ to $G$
10:       for $l \in L$ do
11:         $k'$ := the principal such that $l$ is of the form $k'$ $says$ $\psi$ or $\neg k'$ $says$ $\psi$
12:         $\psi$ := the formula such that $l$ is of the form $k'$ $says$ $\psi$ or $\neg k'$ $says$ $\psi$
13:         if $G$ does not contain a query vertex $k' : \psi$ then
14:           add a query vertex $k' : \psi$ to $G$
15:           add $k' : \psi$ to query_stack
16:         if $l$ is $k'$ $says$ $\psi$ then
17:           add to $G$ a new edge labelled $\mathbf{t}$ from vertex $L$ to vertex $k' : \psi$
18:         if $l$ is $\neg k'$ $says$ $\psi$ then
19:           add to $G$ a new edge labelled $\mathbf{f}$ from vertex $L$ to vertex $k' : \psi$
20:     add to $G$ a new edge from vertex $k : \varphi$ to vertex $L$
21: add the label $\mathbf{u}$ to all query vertices in $G$
22: finished := 0
23: while finished = 0 do
24:   $G_1 := G$
25:   change every $\mathbf{t}$-label on a query vertex in $G_1$ to $\mathbf{u}$
26:   while in $G_1$ there is a query vertex labelled by $\mathbf{u}$ with an edge to a satisfied $says$-literal set vertex do
27:     change every $\mathbf{u}$-label on a query vertex with an edge to a satisfied $says$-literal set vertex to $\mathbf{t}$
28:     $G$ := Make_labelling_of_query_vertices_good($G$)
29:   $G_2 := G$
30:   change every $\mathbf{f}$-label on a query vertex in $G_2$ to $\mathbf{u}$
31:   while in $G_2$ there is a query vertex labelled by $\mathbf{u}$ with an edge to a dissatisfied $says$-literal set vertex do
32:     change every $\mathbf{u}$-label on a query vertex with an edge to a dissatisfied $says$-literal set vertex to $\mathbf{f}$
33:     $G$ := Make_labelling_of_query_vertices_good($G$)
34:   in $G$, change the label on all query vertices that are labelled $\mathbf{u}$ in $G$ and labelled $\mathbf{t}$ in $G_1$ into $\mathbf{t}$
35:   in $G$, change the label on all query vertices that are labelled $\mathbf{u}$ in $G$ and labelled $\mathbf{f}$ in $G_2$ into $\mathbf{f}$
36:   if no changes were made to $G$ in the previous two lines then
37:     finished := 1
38: $V$ := the label on the query vertex $A : \alpha$ in $G$
39: return $V$
```

We now sketch the proof of the equivalence between the $S_{\mathbb{T}}$-based decision procedure and the decision procedure defined in section IV: The only fundamental difference between these two decision procedures is the loop-handling. Step 2) of the $S_{\mathbb{T}}$-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 $S_{\mathbb{T}}$-based decision procedure, which corresponds to making them undecided in the decision procedure defined in section IV.

We now establish that the $S_{\mathbb{T}}$-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 $S_{\mathbb{T}}$ to $(\bot, \top)$, it is now easy to see that the following lemma is sufficient to establish that the $S_{\mathbb{T}}$-based decision procedure always gives the same result as the well-founded semantics:

Lemma 2. *Let $\mathbb{T}$ 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 5 applied to $\mathbb{T}$, 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 5 to G yields a labelling of the queries corresponding to $S_{\mathbb{T}}(B)$.*

Proof. For proving this lemma, it is enough to prove the following four properties, which can be proved straightforwardly:

1) The change in the truth-value labelling of the query vertices of G_1 in line 25 of Algorithm 5 corresponds to changing the belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ to $\bot, \mathcal{Q}_2)$.
2) The change in the truth-value labelling of the query vertices of G_1 in lines 27-28 of Algorithm 5 corresponds to changing the belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ to $(D^*_{\mathbb{T}}(\mathcal{Q}_1, \mathcal{Q}_2)_1, \mathcal{Q}_2)$.
3) The change in the truth-value labelling of the query vertices of G_2 in line 30 of Algorithm 5 corresponds to changing the belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ to $(\mathcal{Q}_1, \top)$.
4) The change in the truth-value labelling of the query vertices of G_2 in lines 32-33 of Algorithm 5 corresponds to changing the belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ to $(\mathcal{Q}_1, D^*_{\mathbb{T}}(\mathcal{Q}_1, \mathcal{Q}_2)_2)$.
5) 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 **t** in correspondence with the distributed belief pair $(\mathcal{Q}_1, \mathcal{Q}_2)$, labelled **f** in correspondence with the distributed belief pair $(\mathcal{Q}_3, \mathcal{Q}_4)$, and labelled **u** otherwise, the resulting labelling corresponds to the distributed belief pair $(\mathcal{Q}_1, \mathcal{Q}_4)$.

□

This completes the proof of Theorem 1.