Theoretical Analysis and Implementation of Abstract Argumentation Frameworks with Domain Assignments

Manuscript Number: IJA-D-23-00102

Giorgos Flouris[1], Theodore Patkos[1], Antonis Bikakis[2], Alexandros Vassiliades[3,4], Nick Bassiliades[3], and Dimitris Plexousakis[1]

[1] Institute of Computer Science, Foundation for Research and Technology, Hellas, Greece
[2] Department of Information Studies, University College London, UK
[3] School of Informatics, Aristotle University of Thessaloniki, Greece
[4] Meaningfy SARLS, Luxembourg
{fgeo,patkos,dp}@ics.forth.gr, a.bikakis@ucl.ac.uk,
alexandros.vassiliadis@meaningfy.ws, nbassili@csd.auth.gr

Abstract. A representational limitation of current argumentation frameworks is their inability to deal with sets of entities and their properties, for example to express that an argument is applicable for a specific set of entities that have a certain property and not applicable for all the others. In order to address this limitation, we recently introduced Abstract Argumentation Frameworks with Domain Assignments (AAFDs), which extend Abstract Argumentation Frameworks (AAFs) by assigning to each argument a domain of application, i.e., a set of entities for which the argument is believed to apply. We provided formal definitions of AAFDs and their semantics, showed with examples how this model can support various features of commonsense and non-monotonic reasoning, and studied its relation to AAFs. In this paper, aiming to provide a deeper insight into this new model, we present more results on the relation between AAFDs and AAFs and the properties of the AAFD semantics, and we introduce an alternative, more expressive way to define the domains of arguments using logical predicates. We also offer an implementation of AAFDs based on Answer Set Programming (ASP) and evaluate it using a range of experiments with synthetic datasets.

1 Introduction

The production and evaluation of argumentative discourse rarely consists of opinions whose claims can precisely be judged as either true or false. There are almost always shades of truthfulness, which make a statement hold in certain situations and not in others. Non-monotonic, case-based, context-aware, defeasible and other forms of reasoning are based, to a smaller or larger extent, on this need to specify the conditions under which a proposition holds or, similarly, to refine the scope within which it can be accounted true. This paper presents a formal method to support such a refinement process using an argumentative formulation.

Both abstract [21] and structured argumentation frameworks [9] enable the revision of the acceptability status of arguments in the light of new evidence or stronger counter-arguments. There is a rich literature on different semantics for deciding the acceptance

of arguments. Yet, these frameworks are challenged when one does not need to decide whether an argument is generally valid or not, but rather to argue on the specific cases where it should be accepted.

Consider the following argument taken from a typical example of commonsense reasoning: "*This fruit is an apple, and since apples are typically red, this fruit is red*". The claim is based on a premise about apples that constitutes common knowledge. Apparently, there are numerous exceptions to this knowledge, such as Granny Smiths, rotten apples etc; nevertheless, an attack on the original argument with the position "*This apple is a Granny Smith and Granny Smiths are green, therefore this apple is green*" should not be read as an attempt to invalidate the argument altogether, but rather to limit the domain of its applicability: apples may still be red in general, and the same argument should still be considered valid in most cases, but when apples belong to a given cultivar, things may be different. Apart from commonsense reasoning, a similar need can be found in other domains, as for instance in negotiation dialogues, especially for multi-party *quid pro quo* conversations, in the legal domain, where the organisation of similar cases within the same argument is a typical practice for modelling legislation, or in security (or any other type of) policies, which specify both the "default" cases and all possible exceptions.

The representation of exceptional cases can be accommodated by non-monotonic logics, but not by argumentation frameworks, in which arguments are *atomic entities*. Thus, unless one creates one argument for each different element of the argument's domain, argumentation frameworks cannot distinguish between an argument about a specific entity and one referring to a set of entities of the same type, and cannot accommodate a notion of "partial acceptance", where an argument is acceptable for some entities only.

Structured argumentation frameworks are often used for modelling scenarios like the above. Under most structured argumentation approaches, such as deductive argumentation [10] or rule-based argumentation (ASPIC$^+$) [37], modelling a domain is based on a set of rules (e.g., "Granny Smiths are typically green") and a set of base facts, that determine, e.g., whether a given object is an apple, Granny Smith or otherwise. Reasoning then proceeds for any given object of interest, and any exceptional cases are considered for this specific object "as they arise" from the given rules and base facts. Interestingly, most structured argumentation frameworks are only used to generate the arguments and their relationships, whereas the reasoning for determining acceptability is still done using some abstract model (typically Dung's AAFs [21]). Thus, these models can handle only part of the reasoning process, namely the argument generation part; acceptability determination is handled by an abstract model.

Motivated by the above, we introduced in [40] a novel argumentation framework called "*Abstract Argumentation Framework with Domain assignments*" (*AAFDs*), in which arguments are abstract, but have a domain of application describing the cases that each argument can be applied to. As with other abstract frameworks, AAFDs include a binary attack relation, whose role is, however, different: instead of invalidating an argument or reducing its strength, it limits its domain of application (called *scope*). This allows capturing and reasoning with exceptional cases and "partial acceptance". Moreover, the concept of domains allows AAFDs to "lift" the domain modelling from

the structured to the abstract level, thus providing an alternative way to perform the entire argumentation pipeline (including both the argument generation and the argument evaluation) using a single model.

In this paper, we extend the theoretical results presented in [40], by providing: (i) a method for partitioning an AAFD according to a partition of its universe (set of elements that can appear in the domain of an argument), which enables computing the extensions of an AAFD by combining the extensions of its parts; and (ii) an alternative, logic-based definition of the domain of an argument, which is more intuitive and expressive, and which enables integrating background knowledge (e.g. from ontologies, knowledge graphs or other knowledge bases) into an AAFD. We also present results on the practical application of the framework; specifically, (iii) an implementation of AAFDs in Answer Set Programming (ASP); (iv) an evaluation of the ASP implementation using a range of experiments with synthetic datasets; and (v) an example application of the ASP implementation for solving a team building problem.

In the rest of the paper, following a discussion of related work in Section 2, we present the definitions of AAFDs and their semantics in Section 3. In Section 4, we present some theoretical results, including a mapping to Abstract Argumentation Frameworks (AAFs), some complexity results, properties of the AAFD semantics, and a method for partitioning an AAFD. In Section 5, we present the ASP implementation, whereas the respective experimental evaluation appears in Section 6. In Section 7, we present the logic-based characterization of AAFDs discussed above. Finally, in Section 8, we conclude the paper and discuss our plans for future work.

2 Related work

Computational argumentation has largely been based on [21], in which an *Abstract Argumentation Framework (AAF)* was defined as a directed graph, i.e., a pair $\langle \mathcal{A}, \mathcal{R} \rangle$ consisting of a (possible infinite) set of arguments $\mathcal{A}$ and a binary attack relation $\mathcal{R}$ on this set. Then, different acceptability semantics for AAFs were defined in terms of extensions, i.e., sets of arguments that can be considered acceptable. In particular, a set $S \subseteq \mathcal{A}$ is:

- *conflict-free* iff it contains no arguments attacking each other, i.e., there is no $a, b \in S$ such that $(a, b) \in \mathcal{R}$
- *admissible* iff it is conflict-free and defends all its elements, i.e., for each argument $b \in \mathcal{A}$ attacking an argument $a \in S$ (that is, $(b, a) \in \mathcal{R}$), there is an argument in $c \in S$ such that $(c, b) \in \mathcal{R}$
- *complete* iff it is admissible and contains all the arguments it defends
- *grounded* iff it is minimal (w.r.t. set inclusion) among the complete extensions
- *preferred* iff it is maximal among the admissible extensions
- *stable* iff it is conflict-free and attacks all the arguments that it does not contain (i.e., for all arguments $b \in \mathcal{A} \setminus S$, there is $a \in S$ such that $(a, b) \in \mathcal{R}$)

Following that paper, various extensions have been proposed, enabling attacks to be directed towards attacks [5], adding support [17], preference [3] or other types of relations [34], adding weights to arguments [2] or attacks [24], associating arguments with

values [8], or imposing hierarchies on arguments [35]. However, little attention has been paid to what we call the *scope* of an argument, i.e., the class of entities that an argument can be applied to, despite the fact that it has been studied in other related areas, such as discourse analysis [38], linguistic argumentation [42] and political argumentation [41].

A recently proposed extension of AAFs allows associating arguments with *topics* [16]. Topics and arguments are semantically interrelated, and the acceptability of an argument depends on the semantic proximity of the arguments that defend it. An important difference with our work is that acceptability is defined in terms of sets of arguments that satisfy certain conditions, so each argument is either accepted or not; in AAFDs, arguments may be partially accepted, i.e., accepted only for a set of entities.

Value-based Argumentation Frameworks (VAF) [8,32], like AAFDs, associate arguments with values. The meaning and the role of these values however is much different in the two frameworks. In VAFs, a value assigned to an argument represents the value that the argument promotes and preferences between values are used to impose an order over the arguments. In AAFDs the elements assigned to an argument represent the entities that the argument can be applied to, and they are inherent parts of the argument.

Structured argumentation frameworks (e.g., ASPIC$^+$ [37], ABA [13], DeLP [26]) provide ways to represent and reason with domain knowledge. For example, Hunter [30] investigates various ways to support non-monotonic reasoning in deductive argumentation [11], while other frameworks leverage (possibly defeasible) domain knowledge expressed in the base logic in the form of rules in order to construct arguments [37,26,13]. Contradictions (possibly explicitly stated) are employed to determine attacks between arguments, which, in turn, allow the determination of acceptable arguments using some abstract argumentation framework.

Non-monotonic inheritance networks [29], which consist of nodes representing individuals or classes, and directed links, representing taxonomic relations, capture the idea of attacks that restrict the scope of arguments. However, they only enable reasoning with taxonomic relations and can be translated to AAFs [22].

Incomplete Abstract Argumentation Frameworks (IAAFs) were originally introduced by Dung in [20]. In applications, incomplete argumentation frameworks may arise as intermediate states in an elicitation process, or when merging different beliefs about the state of an argumentation framework, or in cases where complete information cannot be obtained. In [7] IAAFs were brought back to light as the authors consider two specific models of IAAFs, one focusing on attack incompleteness and the other on argument incompleteness. IAAFs allow various interpretations of the arguments and the attacks that they make, due to the incomplete knowledge we have about them. Our framework indicates upon which elements an argument is applicable, which is different from interpreting arguments and attacks in more than one way, due to missing knowledge.

3 Definitions

3.1 Domain Assignments

In this paper, we equip arguments with a *domain of application*, which corresponds to the elements of a given *universe* that the argument applies to; for example, an argument

about apples refers to a specific set of elements, namely apples, and its validity does not extend to other elements of the universe, e.g., other fruits. The domain of application can be viewed, intuitively, as a way of breaking down an argument into a possibly infinite number of pieces, one for each element of its domain. These pieces do not refer to a logical part of the argument, e.g., a premise, but rather to the elements that the argument (as a whole) applies to, which can be accepted or rejected independently of the rest. This breakdown allows an argument to be attacked only for a specific part of its domain, i.e., some of its pieces, as in the introductory example, where an argument about Granny Smith apples only attacks the argument about apples on the specific subdomain of Granny Smith apples, without affecting its validity for other apples. This idea has the consequence that an attack does not necessarily invalidate an argument totally, and thus an argument can be partially accepted, i.e., accepted for only some elements of its domain.

To formalise these ideas, we use *domain assignments*, which are functions that associate each argument to a *prior* and *posterior* domain of application. The prior domain of application (or simply *domain*) represents the elements of the universe that the argument applies to. Attacks among arguments limit the applicability of the attacked argument, resulting in the argument's posterior domain of application (or simply *scope*), which represents the elements for which the argument is accepted as valid. Formally:

Definition 1 (Domain Assignment). *Consider a non-empty set U, called the* universe*, and a set of arguments $\mathcal{A}$. A* domain assignment of $\mathcal{A}$ to U *is a function that maps each argument to a subset of U, i.e., $D^U : \mathcal{A} \mapsto 2^U$.*

Scopes are the counterparts of extensions in the standard AAF semantics [21]. In particular, an argument a of an AAF is accepted by an extension S if and only if $a \in S$. In AAFDs, acceptance refers to elements of arguments: an element u is accepted for an argument a if and only if $u \in D^U(a)$ for a posterior domain of application (scope) D^U. Note how this captures the intuition of "partial acceptance", as explained above, i.e., the ability to accept an argument only for some of the elements that it applies to.

In the following, we fix some universe U, and omit reference to it for simplicity.

Recall that *Abstract Argumentation Frameworks (AAFs)* are defined in [21] as a pair consisting of a set of arguments and a set of attacks (binary relation) among them. Abstract Argumentation Frameworks with Domain Assignments are AAFs equipped with a domain assignment:

Definition 2 (AAFDs). *An* Abstract Argumentation Framework with Domain assignments*, or* AAFD *for short, is a triple $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$, such that $\mathcal{A}$ is a set of arguments, $\mathcal{R} \subseteq \mathcal{A}^2$ is a binary relation among arguments, and $\overline{D^U} : \mathcal{A} \mapsto 2^U$ is a domain assignment of $\mathcal{A}$ to U, called the* prior domain assignment *(or simply domain).*

3.2 Semantics for AAFDs: Intuition and Desiderata

In Abstract Argumentation Frameworks [21], semantics are defined through extensions, which are subsets of the original set of arguments that satisfy specific properties. To allow partial acceptance, our analysis needs to be more fine-grained, and should concern

the arguments' domains, rather than the arguments themselves. In particular, the semantics of an AAFD is defined as a domain assignment (the scope) that determines the subsets of the arguments' domains that satisfy certain properties, giving rise to semantics analogous to the classical ones (admissible, complete, etc.).

Before defining formally the semantics of an AAFD, we provide some relevant desiderata, inspired by postulates and principles defined in other related contexts (e.g., [6,23]).

The first requirement is about the relationship between the domain and the scope of the argument. In particular, we argue that the scope of an argument must always be a subset of its domain. This is reasonable, because the applicability of an argument is an inherent property of the argument, determined by the modeller based on its logical content; therefore, the argumentation framework (and its attacks) cannot increase it. This gives rise to the *domain capping constraint (DC)*.

Further, the domain of an argument should not be arbitrarily reduced. For every element "missing" from the domain of an argument, there should be a valid reason, i.e., some external attack. In other words, if a subset of the domain of an argument receives no attack, it should be part of the argument's scope. This gives rise to the *scope maximality constraint (SM)*. Note that a corollary of (DC) and (SM) combined is that an argument receiving no attack should be fully accepted, i.e., its scope should be equal to its domain.

The third requirement ensures that arguments are consistent to each other with respect to their scope. For example, if a attacks b, and u is in the domain of both a, b, then u cannot be in the scope of both arguments. This is called the *scope consistency constraint (SC)*.

Another interesting intuition stems from considering elements that always appear "together" in the arguments' domains. Such elements are, in a sense, "indistinguishable", and there seems to be no reason to differentiate among them. In other words, elements that always appear together in the arguments' domains, should also appear together in their scope. This is the *equal treatment constraint (ET)*.

To formalise this constraint, we will need to define the concept of elements that always appear "together":

Definition 3 (Indistinguishable). *Two elements $u_1, u_2 \in U$ are called* indistinguishable*, if and only if, for all $a \in \mathcal{A}$, $u_1 \in \overline{D^U}(a)$ if and only if $u_2 \in \overline{D^U}(a)$. We write $u_1 \sim u_2$ to denote that u_1, u_2 are indistinguishable.*

It is trivial to see that the relation $\sim$ is an equivalence relation (i.e., reflexive, symmetric and transitive), and thus breaks down U into disjoint equivalence classes. We denote by $\mathcal{C} = \{C_i\}$ the equivalence classes generated by $\sim$.

Using Definition 3, we can now formally define the above constraints:

DC $\forall a \in \mathcal{A}, D^U(a) \subseteq \overline{D^U}(a)$
SM $\forall a \in \mathcal{A}$ and $\forall u \in \overline{D^U}(a) \setminus D^U(a)$,
$\exists b \in \mathcal{A}$ such that $(b, a) \in \mathcal{R}$ and $u \in \overline{D^U}(b)$
SC $\forall a, b \in \mathcal{A}$ and $\forall u \in U$, $(a, b) \in \mathcal{R}$ implies that $u \notin D^U(a) \cap D^U(b)$
ET $\forall a \in \mathcal{A}$ and $\forall u_1, u_2 \in U$ such that $u_1 \sim u_2$, it holds that $u_1 \in D^U(a)$ if and only if $u_2 \in D^U(a)$

3.3 Semantics for AAFDs: formal definitions

To define our semantics, we first introduce the notion of *limiting the domain of application*: given a set of arguments $\mathcal{A}$, and two domain assignments D_1^U, D_2^U for $\mathcal{A}$, we say that D_1^U *limits* D_2^U, denoted by $D_1^U \sqsubseteq D_2^U$, if and only if $D_1^U(a) \subseteq D_2^U(a)$ for all $a \in \mathcal{A}$.

In the following, we define various classes of scope, which give rise to the respective semantics. In all the following definitions, we assume a fixed AAFD $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$, and some scope $D^U : \mathcal{A} \mapsto 2^U$. As an example, we use the simple AAFD depicted in Figure 1, where $U = \{x, y, z\}$, $\mathcal{A} = \{a, b\}$, $\mathcal{R} = \{(a, b), (b, a)\}$ and $\overline{D^U}(a) = \{x, y\}$, $\overline{D^U}(b) = \{x, y, z\}$. If we disregard the domains of a and b, the argument graph is the same as the one that was used by Dung to represent the famous Nixon's diamond problem, with a (in Dung's paper) standing for the argument "Nixon is anti-pacifist since he is a republican", and b standing for "Nixon is a pacifist since he is a Quaker" [21]. Here, a and b represent two more general arguments: "all republicans are anti-pacifists", and "all Quakers are pacifists", respectively. Nixon is represented by one of the elements in the domain of both arguments, for example, x. The other two elements represent other people that we may want to reason about, e.g., y may represent Herbert Hoover who, like Nixon, was both a republican and a Quaker, while z may represent John Hickenlooper, a Quaker that is a member of the Democratic Party.

We start with the notion of compliant scope:

Definition 4 (Compliant scope). *The scope* D^U is compliant *if and only if* $D^U \sqsubseteq \overline{D^U}$.

A scope that is compliant simply guarantees that no arbitrary assignments are made as described by the DC requirement (see also Proposition 7). For this example framework, the scope $D_1^U(a) = \{x, y\}$, $D_1^U(b) = \{y, z\}$ (in the context of the Nixon's diamond problem, this would mean that Nixon and Hoover are anti-pacifists, and Hoover and Hickenlooper are pacifists) is compliant.

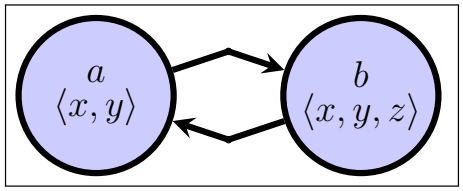

Fig. 1: An example AAFD.

Definition 5 (Conflict-free). *The scope* D^U *is* conflict-free *if and only if it is compliant, and for any* $a, b \in \mathcal{A}$, $u \in U$, *if* $(a, b) \in \mathcal{R}$ *then* $u \notin D^U(a) \cap D^U(b)$.

Conflict-freeness ensures that the SC requirement is satisfied (see Proposition 7). Returning to the example of Figure 1, we note that D_1^U is not conflict-free, but $D_2^U(a) = \{x\}$, $D_2^U(b) = \{y, z\}$ is. *Acceptability* is defined as follows:

Definition 6 (Acceptability). *For $a \in \mathcal{A}, u \in U$, such that $u \in \overline{D^U}(a)$, the pair (a, u) is* acceptable with respect to D^U *if and only if, whenever $(b, a) \in \mathcal{R}$ and $u \in \overline{D^U}(b)$, there exists some $c \in \mathcal{A}$ such that $(c, b) \in \mathcal{R}$ and $u \in D^U(c)$.*

An argument $a \in A$ is called *fully acceptable with respect to D^U* iff (a, u) is acceptable with respect to D^U for all $u \in \overline{D^U}(a)$ and $\overline{D^U}(a) \neq \emptyset$. It is called *non-acceptable with respect to D^U* iff there is no $u \in \overline{D^U}(a)$ such that (a, u) is acceptable with respect to D^U. It is called *partially acceptable with respect to D^U* if it is neither fully acceptable, nor non-acceptable with respect to D^U. Using Definition 6, we can now define admissibility in AAFDs as follows:

Definition 7 (Admissible). *The scope D^U is* admissible *if and only if it is conflict-free, and (a, u) is acceptable with respect to D^U, for all $a \in \mathcal{A}$, $u \in D^U(a)$.*

Continuing the example of Figure 1, we note that the scope $D_3^U(a) = \{x\}, D_3^U(b) = \{z\}$ is admissible.

Definition 8 (Complete). *The scope D^U is* complete *if and only if it is admissible, and for any $a \in \mathcal{A}$, $u \in \overline{D^U}(a)$, if (a, u) is acceptable with respect to D^U, then $u \in D^U(a)$.*

In the AAFD of Figure 1, the scope $D_4^U(a) = \{x, y\}, D_4^U(b) = \{z\}$ is complete. Complete scopes satisfy the SM requirement (see Proposition 7).

Definition 9 (Grounded). *The scope D^U is* grounded *if and only if it is a minimal (with respect to $\sqsubseteq$) complete scope.*

Definition 10 (Preferred). *The scope D^U is* preferred *if and only if it is a maximal (with respect to $\sqsubseteq$) admissible scope.*

Definition 11 (Stable). *The scope D^U is a* stable scope *if and only if it is conflict-free, and for any $a \in \mathcal{A}$, $u \in U$, if $u \in \overline{D^U}(a) \setminus D^U(a)$, then there exists $b \in \mathcal{A}$ such that $(b, a) \in \mathcal{R}$ and $u \in D^U(b)$.*

To illustrate the above definitions, let us consider again the AAFD in Figure 1, and the following assignments:

- $D_5^U(a) = \{x, y\}, D_5^U(b) = \{z\}$.
- $D_6^U(a) = \{x\}, D_6^U(b) = \{y, z\}$.
- $D_7^U(a) = \emptyset, D_7^U(b) = \{z\}$.
- $D_8^U(a) = \emptyset, D_8^U(b) = \{x, y, z\}$.

We can easily show that all the above assignments are complete. The only grounded one is D_7^U, whereas D_5^U, D_6^U, D_8^U are preferred and stable.

None of the above types of scope captures the ideas behind the ET requirement described in Section 3.2. The following definition achieves that, using the notion of indistinguishable elements (Definition 3):

Definition 12 (Canonical). *The scope D^U is* canonical *if and only if for all $a \in \mathcal{A}$, $u_1, u_2 \in U$, whenever $u_1 \sim u_2$ and $u_1 \in D^U(a)$, it follows that $u_2 \in D^U(a)$.*

Looking at the previous examples, we note that D_4^U, D_5^U, D_7^U and D_8^U are canonical, whereas $D_1^U, D_2^U, D_3^U, D_6^U$ are not. Note that canonical semantics are orthogonal to other semantics, so any scope that is, e.g., admissible, complete, etc., may (or may not) also be canonical.

In the following, we use the following shorthands for the semantics introduced above: $\mathbf{cm}$ for compliant, $\mathbf{cf}$ for conflict-free, $\mathbf{ad}$ for admissible, $\mathbf{co}$ for complete, $\mathbf{pr}$ for preferred, $\mathbf{gr}$ for grounded, $\mathbf{st}$ for stable and $\mathbf{ca}$ for canonical. We use σ as a catch-all symbol to indicate any of the above; for example, we write $\mathbf{co}$-scope to refer to a complete scope, and σ-scope to refer to a scope of the type denoted by σ.

Following common practice in the argumentation literature, we can also define *credulous* and *skeptical* justification based on the available semantics [4]. However, due to the nature of AAFDs, these notions apply to elements of arguments, not to arguments alone, and are based on the scopes (which are the counterpart of extensions in the standard AAFs). Formally:

Definition 13. *Consider an AAFD $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$, some scope semantics σ, an element $u \in U$ and some argument $a \in \mathcal{A}$. We say that u is* credulously justified *(or* credulously accepted*) if and only if there exists some σ-scope D^U such that $u \in D^U(a)$. We say that u is* skeptically justified *(or* skeptically accepted*) if and only if $u \in D^U(a)$ for all σ-scopes D^U and there exists at least one σ-scope D^U.*

4 Formal properties and results for AAFDs

4.1 Mapping AAFDs to AAFs

Interestingly, it can be shown that the semantics of AAFDs are fully compatible with the classical semantics of AAFs [21], i.e., an AAFD can be viewed as an AAF (and vice-versa, of course, e.g., when U is a singleton set). On the other hand, AAFDs provide a much more compact and intuitive representation. Below, we describe two different ways to map AAFDs to AAFs.

Natural mapping. For the first mapping, we assume that each element in the domain of an argument is spawning an argument in the AAF (creating also attacks in the natural way). In other words, each element $u \in \overline{D^U}(a)$, along with the argument a itself (as a pair, a^u) are mapped into an argument in the AAF (see also Figure 2). Formally, the *natural mapping* is a mapping Φ_{nat}, which maps an AAFD $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$ to an AAF $\langle \mathcal{A}_*, \mathcal{R}_* \rangle = \Phi_{nat}(\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle)$, as follows:

- $\mathcal{A}_* = \{a^u \mid a \in \mathcal{A}, u \in \overline{D^U}(a)\}$
- $\mathcal{R}_* = \{(a^u, b^u) \mid a^u, b^u \in \mathcal{A}_*, (a, b) \in \mathcal{R}\}$

For simplicity, we abuse notation, and denote by $\Phi_{nat}(D^U) = \{a^u \mid a \in \mathcal{A}, u \in D^U(a)\}$, where D^U is a compliant scope. In words, $\Phi_{nat}(D^U)$ returns all elements a^u such that u is in the scope of a (according to D^U).

The assumption for D^U being a compliant scope is crucial for defining $\Phi_{nat}(D^U)$. Indeed, if D^U was not compliant, then it would force us to spawn elements of the form a^u, where $u \notin \overline{D^U}(a)$; such elements do not appear in $\mathcal{A}_*$.

The following proposition shows the equivalence in the semantics of the two representations:

Proposition 1. *Take an AAFD $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$, the AAF $\langle \mathcal{A}_*, \mathcal{R}_* \rangle = \Phi_{nat}(\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle)$, and a compliant scope D^U in $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$. Then, for $\sigma \in \{\mathbf{cf}, \mathbf{ad}, \mathbf{co}, \mathbf{gr}, \mathbf{pr}, \mathbf{st}\}$, D^U is a σ-scope in $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$ if and only if $\Phi_{nat}(D^U)$ is a σ-extension in $\langle \mathcal{A}_*, \mathcal{R}_* \rangle$.*

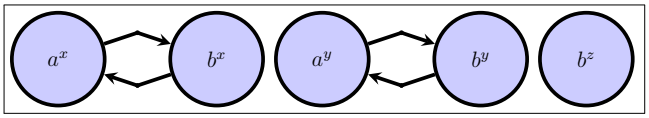

Fig. 2: Applying Φ_{nat} on the AAFD of Figure 1

Natural mapping with equivalences. Although Φ_{nat} achieves the required effect of mapping an AAFD into an AAF with equivalent semantics, it is not efficient because the size of the resulting AAF is highly dependent on the size of U, which is expected to be large (or even infinite). We show here a more sophisticated mapping, which leads to an AAF whose size only depends on the number of arguments in AAFD, but not on the size of U. The idea stems from the notion of indistinguishable elements (Definition 3), and applies only for compliant and canonical scopes.

The *natural mapping with equivalences* $\Phi_{[nat]}$ is very similar to Φ_{nat}, except that it creates, for each $a \in \mathcal{A}$ and for each equivalence class C_i, just one argument in the AAF. In other words, if $a \in \mathcal{A}$, $u_1 \sim u_2$ (say $u_1, u_2 \in C_1$) and $u_1, u_2 \in \overline{D^U}(a)$, then, instead of creating two arguments in the AAF (namely a^{u_1}, a^{u_2}), $\Phi_{[nat]}$ creates just one (a^{C_1}). Figure 3 shows the application of $\Phi_{[nat]}$ on the AAFD of Example 1, for the equivalence classes $C_1 = \{x, z\}$, $C_2 = \{y\}$.

Formally, we define a mapping $\Phi_{[nat]}$, which maps an AAFD $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$ to an AAF $\langle \mathcal{A}_*, \mathcal{R}_* \rangle = \Phi_{[nat]}(\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle)$, as follows:

- $\mathcal{A}_* = \{a^C \mid a \in \mathcal{A}, C \in \mathcal{C}, C \subseteq \overline{D^U}(a)\}$
- $\mathcal{R}_* = \{(a^C, b^C) \mid a^C, b^C \in \mathcal{A}_*, C \in \mathcal{C}, (a, b) \in \mathcal{R}\}$

Again, we abuse notation, and denote by $\Phi_{[nat]}(D^U) = \{a^C \mid a \in \mathcal{A}, C \subseteq D^U(a)\}$, where D^U is a compliant and canonical scope. As before, this allows us to model the scope of AAFD arguments in the AAF representation. Note that, in the above definition of $\Phi_{[nat]}(D^U)$, the assumption for D^U being a compliant and canonical scope is crucial. For compliance, the reason is similar to the respective reason given in the description of the natural mapping Φ_{nat}. With respect to canonical, suppose that D^U is not canonical and take some $C \in \mathcal{C}$, $u_1, u_2 \in C$, such that $u_1 \in D^U(a)$, $u_2 \notin D^U(a)$. Then, it is not clear whether a^C (which essentially represents both u_1 and u_2 as candidates for inclusion in the scope of a) should be in $\Phi_{[nat]}(D^U)$.

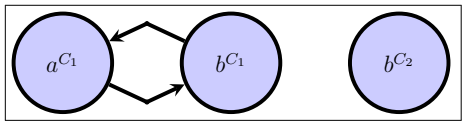

Fig. 3: Applying $\Phi_{[nat]}$ on the AAFD of Figure 1

The following result shows that $\Phi_{[nat]}$ preserves the semantics during the mapping of an AAFD to an AAF; note, however, that this result applies only when we restrict ourselves to the realm of canonical scopes:

Proposition 2. *Take an AAFD $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$, the AAF $\mathcal{F}_* = \langle \mathcal{A}_*, \mathcal{R}_* \rangle$, where $\mathcal{F}_* = \Phi_{[nat]}(\mathcal{F})$, and a compliant and canonical scope D^U in $\mathcal{F}$. Then, for $\sigma \in \{\mathbf{cf}, \mathbf{ad}, \mathbf{co}, \mathbf{gr}, \mathbf{pr}, \mathbf{st}\}$, D^U is a σ-scope in $\mathcal{F}$ if and only if $\Phi_{[nat]}(D^U)$ is a σ-extension in $\mathcal{F}_*$.*

4.2 Complexity results

Next, we consider the problem of determining the complexity of reasoning with AAFDs, as well as the size benefits that the more compact representation of AAFDs provides, compared to its respective AAF (through the Φ_{nat} and $\Phi_{[nat]}$ mappings).

We first address the problem of how to represent AAFs and AAFDs, as well as the respective sizes. For AAFs, the result is trivial: the size of $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ is $|\mathcal{F}| = |\mathcal{A}| + |\mathcal{R}| \leq |\mathcal{A}| + |\mathcal{A}|^2 = O(|\mathcal{A}|^2)$. For AAFDs the respective result is given in the following proposition:

Proposition 3. *Consider an AAFD $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$ defined over universe U, and let $\mathcal{C}$ be the set of equivalence classes determined by the equivalence relation $\sim$ over $\mathcal{F}$. Then:*

1. $|\mathcal{F}| \leq |\mathcal{A}| + |\mathcal{A}|^2 + |\mathcal{A}| \cdot |U|$
2. $|\mathcal{F}| = O(|\mathcal{A}|^2 + |\mathcal{A}| \cdot |U|)$
3. $|\mathcal{C}| \leq \min\{2^{|\mathcal{A}|}, |U|\}$

Using the above result, Proposition 4 provides bounds (both standard and asymptotic) for the size of an AAF that corresponds to an AAFD (under Φ_{nat}, $\Phi_{[nat]}$):

Proposition 4. *Consider an AAFD $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$. Set $\mathcal{F}_1 = \Phi_{nat}(\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle)$, $\mathcal{F}_2 = \Phi_{[nat]}(\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle)$. Then:*

1. $|\mathcal{F}_1| \leq |\mathcal{A}| \cdot |U| + |\mathcal{A}|^2 \cdot |U|^2$
2. $|\mathcal{F}_1| = O(|\mathcal{A}|^2 \cdot |U|^2) = O(|\mathcal{F}|^2)$
3. $|\mathcal{F}_2| \leq |\mathcal{A}| \cdot 2^{|\mathcal{A}|} + |\mathcal{A}|^2 \cdot 2^{2 \cdot |\mathcal{A}|}$
4. $|\mathcal{F}_2| \leq |\mathcal{A}| \cdot |U| + |\mathcal{A}|^2 \cdot |U|^2$
5. $|\mathcal{F}_2| = O(|\mathcal{A}|^2 \cdot \min\{2^{2 \cdot |\mathcal{A}|}, |U|^2)\}) = O(|\mathcal{F}|^2)$
6. $|\mathcal{F}_2| \leq |\mathcal{F}_1|$

Several interesting observations can be made from Proposition 4. First, the AAFD representation is typically more compact than its respective AAF (see results #2, #5), as the latter can be quadratic in size in the worst-case scenario (under both mappings) with respect to the former. Note that this is only a worst-case scenario: for example, one could create an AAFD where the elements are assigned in arguments in such a way that the respective attacks do not materialise (i.e., arguments attacking each other having disjoint elements). Such a structure would create a small AAF, as the non-materialised attacks will not be represented in the result. If such non-materialised attacks are numerous enough, an AAFD may end up being larger than its respective AAF. On the other hand, if we don't have such "bogus" attacks in the AAFD, then the AAFD is a more compact representation, because the respective AAF will have as many arguments as the mappings of $\overline{D^U}$ in the AAFD (typically many more than the AAFD's arguments), and its attacks will correspondingly be much more numerous.

Another interesting observation stems from results #3 and #4, and the fact that we typically expect $|\mathcal{A}|$ to be much smaller than $|U|$, as the elements in question are usually expected to be more numerous than the rules (arguments) that apply to them. If it is the case that $2^{|\mathcal{A}|} \leq |U|$, then the bound given by result #3 is tighter than the one given by #4, and thus the worst-case size of $\Phi_{[nat]}(\mathcal{F})$ does not depend on U. On the other hand, when U is not as large, we can still employ the bound of result #4, which is identical to the one given for Φ_{nat} (result #1). The same observations apply for the asymptotic sizes. A corollary of the above analysis is also the fact that $\Phi_{[nat]}$ is at least as space-efficient as Φ_{nat} in all cases (result #6). The following important corollary follows also from result #3 of Proposition 4:

Corollary 1. *Take some $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$. If $\mathcal{A}, U$ are finite, then $\Phi_{nat}(\mathcal{F})$ is finite. If $\mathcal{A}$ is finite, then $\Phi_{[nat]}(\mathcal{F})$ is finite.*

With regards to time complexity, Proposition 5 below shows that there is a quadratic-time reduction of any reasoning problem in AAFDs to its respective problem in AAFs (as expected, by the results of Proposition 4):

Proposition 5. *For any given reasoning problem in AAFDs, there is a polynomial-time (quadratic) reduction to the respective reasoning problem in AAFs.*

It should be noted that the above analysis regarding the space and time complexity applies for the simple representation of AAFDs, where domains are represented through the enumeration of their elements. More sophisticated representations (e.g., through predicates or other methods – see Section 7) may have a much smaller representational footprint, but may incur additional computational overheads (to determine the domains, as needed). Further analysis would require specific assumptions on the type of predicates/formulas used for the representation, and is omitted.

4.3 Properties of AAFD semantics

The equivalence between the semantics of AAFDs and AAFs (Proposition 1) means that most of the results that have appeared in the literature regarding AAFs can be trivially applied to AAFDs as well (e.g., on the existence, multiplicity, and relation of

different types of scope). In particular, $\mathbf{cf}$, $\mathbf{ad}$, $\mathbf{co}$, $\mathbf{gr}$ and $\mathbf{pr}$-scopes always exist, the $\mathbf{gr}$-scope is unique, a $\mathbf{st}$-scope is also a $\mathbf{pr}$-scope, a scope that is $\mathbf{pr}$ or $\mathbf{gr}$ is also $\mathbf{co}$, a $\mathbf{co}$-scope is $\mathbf{ad}$, and an $\mathbf{ad}$-scope is $\mathbf{cf}$. Also, a $\mathbf{cf}$-scope is $\mathbf{cm}$, by definition. The proof of these results follows directly from Proposition 1 and the respective results in [21] and elsewhere. We can also show the following, which is specific for AAFDs:

Proposition 6. *In any AAFD, there exists at least one* $\mathbf{cm}$*-scope and at least one* σ*-scope that is also a* $\mathbf{ca}$*-scope, for* $\sigma \in \{\mathbf{cm}, \mathbf{cf}, \mathbf{ad}, \mathbf{co}, \mathbf{gr}, \mathbf{pr}\}$*. Also, if there exists a* $\mathbf{st}$*-scope, then there exists a* $\mathbf{st}$*-scope that is also a* $\mathbf{ca}$*-scope.*

The following proposition shows how the various semantics defined in Section 3.3 meet the requirements described in Section 3.2:

Proposition 7. *Consider an AAFD* $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$ *and a scope* D^U*. Then:*

- *For* $\sigma \in \{\mathbf{cm}, \mathbf{cf}, \mathbf{ad}, \mathbf{co}, \mathbf{pr}, \mathbf{gr}, \mathbf{st}\}$*, if* D^U *is a* σ*-scope, then it satisfies DC.*
- *For* $\sigma \in \{\mathbf{co}, \mathbf{pr}, \mathbf{gr}, \mathbf{st}\}$*, if* D^U *is a* σ*-scope, then it satisfies SM.*
- *For* $\sigma \in \{\mathbf{cf}, \mathbf{ad}, \mathbf{co}, \mathbf{pr}, \mathbf{gr}, \mathbf{st}\}$*, if* D^U *is a* σ*-scope, then it satisfies SC.*
- *If* D^U *is a* $\mathbf{ca}$*-scope, then it satisfies ET.*

Corollary 2. *Consider an AAFD* $\langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$ *and* $\sigma \in \{\mathbf{co}, \mathbf{gr}, \mathbf{pr}\}$*. Then:*

- *There exists a scope* D^U *that is* $\mathbf{ca}$*-*σ*.*
- *If* D^U *is* $\mathbf{ca}$*-*σ*, then it satisfies DC, SM, SC, ET.*
- *If* D^U *is* $\mathbf{ca}$*-*$\mathbf{st}$*, then it satisfies DC, SM, SC, ET.*

From Corollary 2, it follows that a scope that is $\mathbf{ca}$-$\mathbf{co}$, $\mathbf{ca}$-$\mathbf{gr}$, or $\mathbf{ca}$-$\mathbf{pr}$ always exists, and it satisfies all requirements set in Subsection 3.2. Thus, although suitability of semantics is application-dependent, we argue that $\mathbf{ca}$-$\mathbf{co}$, $\mathbf{ca}$-$\mathbf{gr}$ and $\mathbf{ca}$-$\mathbf{pr}$-scopes seem more plausible for most applications. The same is true for $\mathbf{ca}$-$\mathbf{st}$-scopes, when they exist.

4.4 Partitions of AAFDs

An important observation regarding AAFDs is that the computation of the semantics for some element (or subset of elements) of the universe is independent of the other elements. Therefore, one can "break down" the computation of an AAFD's semantics by considering partitions of the universe U, and restricting the AAFD over these partitions. Then, by combining the resulting semantics of each partition, we can get the semantics of the original AAFD. Note that any partition of U would do; in the extreme case, each element of the universe can be treated individually.

To understand this, consider the AAFD shown in the top-left of Figure 4. We can break down (partition) U into its elements, and compute the respective scope for each, as visualised in the other boxes of Figure 4. Then, we can combine the semantics of the different restrictions, by taking all combinations; the result will be the scope of the original AAFD.

For example, consider the scope: $D^U(a) = \{z\}, D^U(b) = \{w, v\}, D^U(c) = \{y\}, D^U(d) = \{w, v\}$. It can be verified that this is a complete scope. It would also result from the combination of the individual partitions, because:

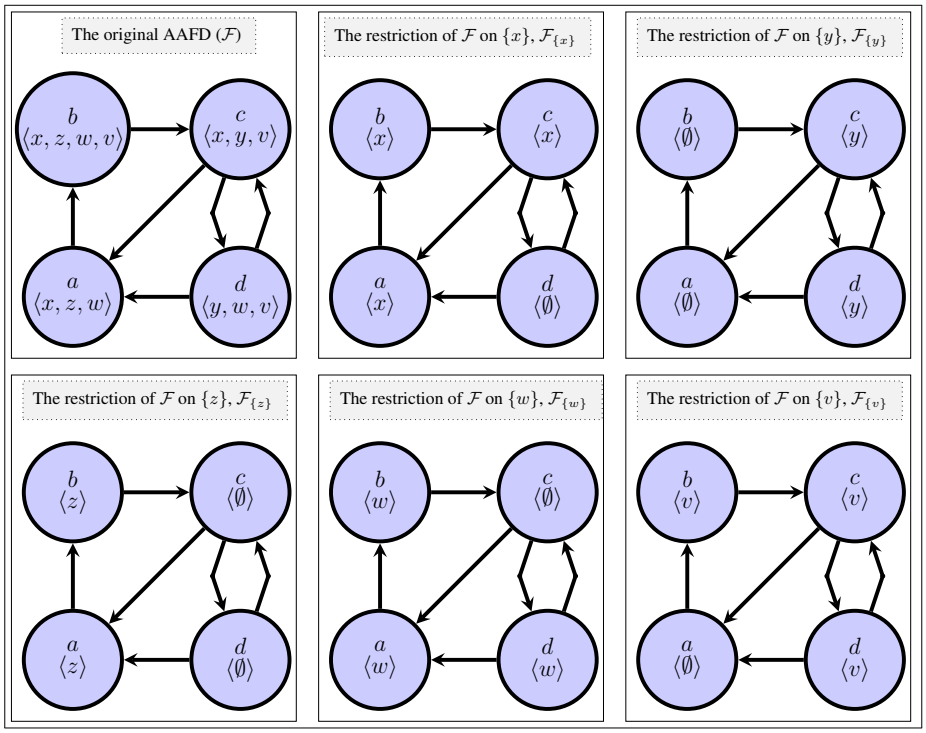

Fig. 4: A complex AAFD ($\mathcal{F}$), and its canonical partition

- In $\mathcal{F}_{\{x\}}$, the scope that assigns x to no argument is complete.
- In $\mathcal{F}_{\{y\}}$, the scope that assigns y to c is complete.
- In $\mathcal{F}_{\{z\}}$, the scope that assigns z to a is complete.
- In $\mathcal{F}_{\{w\}}$, the scope that assigns w to b, d is complete.
- In $\mathcal{F}_{\{v\}}$, the scope that assigns v to b, d is complete.

Similarly, if you take any other combination of **co**-scopes for the restricted AAFDs and combine them, you get a complete extension for $\mathcal{F}$ (and vice-versa); and the same holds for all types of semantics. This idea is formalised next.

Definition 14. *Consider an AAFD $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U}\rangle$, and a set $V \subseteq U$. The* restriction *of $\mathcal{F}$ on V is the AAFD $\mathcal{F}_V = \langle \mathcal{A}_V, \mathcal{R}_V, \overline{D^V}\rangle$, whose universe is V, and which is defined as follows:*

- $\mathcal{A}_V = \mathcal{A}$
- $\mathcal{R}_V = \mathcal{R}$
- *For all $a \in \mathcal{A}$, $\overline{D^V}(a) = \overline{D^U}(a) \cap V$*

Now, we define the partition of an AAFD, which consists of breaking down the universe into disjoint pieces, and taking the restriction for each:

Definition 15. *Consider an AAFD $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$, and a family of sets $\mathcal{V} = \{V_i\}$, such that $V_i \subseteq U$ for all i. Set $\mathcal{F}_{V_i}$ the restriction of $\mathcal{F}$ on V_i. Then, the family $\{\mathcal{F}_{V_i}\}$ is called a* partition *of $\mathcal{F}$ if and only if $\bigcup_i V_i = V$ and $V_i \cap V_j = \emptyset$ for all $i \neq j$.*

Given an AAFD $\mathcal{F}$ and its corresponding equivalence relation ($\sim$), and the set of equivalence classes that $\sim$ generates (say $\{C_i\}$), the family $\{\mathcal{F}_{C_i}\}$ is a partition of $\mathcal{F}$ that we will call the *canonical partition*. Another interesting partition is the one in which all V_i are singleton sets, i.e., $\{\mathcal{F}_{\{u\}} \mid u \in U\}$; this is called the *atomic partition*. Note that the partition shown in Figure 4 is an atomic partition, but also a canonical one.

Next, we provide a method to combine the scopes of the AAFDs in a partition, to create the "combined scope":

Definition 16. *Given a family of functions $f_i : X \mapsto 2^{Y_i}$, we define the* concatenation of f_i *as the function $f : X \mapsto 2^{\bigcup_i Y_i}$ such that $f(x) = \bigcup_i f_i(x)$. We denote f by $\sqcup_i f_i$.*

Note that scopes are essentially mappings (i.e., functions) returning sets of elements. Thus, the concatenation operator can be applied on them, and corresponds to the "combination" (mentioned informally above) of the scopes of the AAFDs that compose the partition of a larger AAFD. More formally, the following theorem holds:

Proposition 8. *Consider an AAFD $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$ and a partition $\{\mathcal{F}_{V_i}\}$ of $\mathcal{F}$. Then, the following hold:*

1. *For $\sigma \in \{\mathbf{cm}, \mathbf{cf}, \mathbf{ad}, \mathbf{co}, \mathbf{pr}, \mathbf{gr}, \mathbf{st}\}$ it holds that D^{V_i} is a σ-scope in $\mathcal{F}_{V_i}$ for all i if and only if $\sqcup_i D^{V_i}$ is a σ-scope in $\mathcal{F}$*
2. *If $\{\mathcal{F}_{V_i}\}$ is the canonical partition, then D^{V_i} is a $\mathbf{ca}$-scope in $\mathcal{F}_{V_i}$ for all i if and only if $\sqcup_i D^{V_i}$ is a $\mathbf{ca}$-scope in $\mathcal{F}$*

Proposition 8 essentially captures the intuition mentioned in the beginning of this subsection: we can decide whether a scope is a σ-scope (for any semantics σ), by checking whether its components generated by any given partition are σ-scope. Conversely, we can construct a σ-scope by identifying σ-scopes of smaller AAFDs (the partitions of the original). The latter observation means that the above result can be leveraged in different ways when it comes to the implementation of AAFD semantics for more efficient implementations.

One possible way is to consider the atomic partition. Under this viewing, each of the restricted AAFDs has a singleton universe, so it is essentially an AAF, and standard, optimized algorithms for computing each semantics can be used (see [18] for a related survey). Then, the results of these algorithms can be composed to compute the actual σ-scopes of the original AAFD.

Another way is to use the canonical partition, which also guarantees that the results will be canonical (according to the second case of Proposition 8). On the other hand, a canonical partition is not necessarily an atomic one, so we need to resort to implementations specifically tailored for AAFDs (or, we could use the trick above with the atomic partition, to reduce each computation to the computation of AAF semantics).

5 Implementation

In this section, we present an implementation of the different AAFD semantics using a declarative programming language, namely the Answer Set Programming (ASP [33]) formalism. We further discuss the encoding of a problem that is often met in various domains, in order to showcase the simplicity in modelling both theoretical and practical notions using AAFDs. A Web interface is also available for users to run custom examples[1].

5.1 An Encoding of AAFD Semantics

The implementation of the different AAFD semantics was driven by our aim to generate a simple and intuitive, rather than an efficient, encoding. As such, we chose ASP, a highly expressive declarative programming language, which enables the encoding of the semantics to be very close to the formal definitions, and the formulation of application domains and queries to become straightforward. The fact that ASP can be executed by very efficient reasoners, such as Clingo [27] and Asprin[2] [14] is a bonus; nevertheless, if efficiency is the target, one can develop custom programming techniques or rely on solvers properly adapted for efficient handling of argumentation frameworks, which might result in better performance, such as ASPARTIX [25], pyglaf [1], μ-toksia [36] or ConArg [12].

For our programs, a problem instance needs to be modelled following a specific format, in order to be properly interpreted by the AAFD encodings. Specifically, arguments definition, elements definition, domain assignments, and attack relations need to be modelled as shown next:

```
argument(arg1).
argument(arg2).
...
element(elem1).
element(elem2).
...
domain(argXX, elemYY).
...
attacks(argXX, argZZ).
```

That is, the $argument/1$ and $element/1$ unary predicates, and the $domain/2$ and $attacks/2$ binary predicates can be used to populate a particular domain and to axiomatise the attack relations among arguments. For instance, $attacks(arg1, arg2)$ defines an attack from argument $arg1$ to argument $arg2$, whereas $domain(arg1, elemA)$ indicates that the element $elemA$ is assigned to the domain of argument $arg1$.

We now explain how we compute the different AAFD semantics using ASP (the code is also available in Appendix B). For all semantics, the program first generates

[1] http://139.91.183.45:8060/

[2] https://potassco.org/

all possible scopes, i.e., all possible assignments of elements to arguments, using the following choice rule (note that words starting with a capital letter denote variables):

```
{scope(A, U) : element(U)} :- argument(A).
```

Given the argument and element instances defined for a particular domain, this rule generates all possible *scopes*, i.e., all possible assignments of elements to arguments.

Then, using the constraint shown next, the program eliminates all non-compliant scopes, i.e., all scopes that assign elements to arguments that are not included in their domain; in other words, with this constraint in place, the program implements the definition of *compliant scopes* (Def. 4):

```
:- scope(A, U), not domain(A, U).
```

The following constraint further eliminates any scope that assigns the same element to two arguments that are in conflict, resulting in the set of *conflict-free scopes* (Def. 5):

```
:- attacks(A1, A2), scope(A1, U), scope(A2, U).
```

In order to implement the **ad**-semantics, additionally to the above clauses, we use the following rules to define when an assignment is *acceptable* (as per Def. 6):

```
argumentGetsAttackOn(A, U, B) :-
        domain(A, U), domain(B, U),
        attacks(B, A).

defendedFrom(A, U, B) :-
        argumentGetsAttackOn(A, U, B),
        attacks(C, B), scope(C, U).

notAcceptable(A, U) :-
        argumentGetsAttackOn(A, U, B),
        not defendedFrom(A, U, B).

acceptable(A, U) :-
    domain(A, U), not notAcceptable(A, U).
```

The intuition is that an argument should not be attacked on its assigned element, or it should be defended from *any* attacks it receives on its assigned element. We also use the following constraint to filter out scopes containing assignments that are not acceptable, implementing the definition of *admissible scopes* (Def. 7):

```
:- scope(A, U), not acceptable(A, U).
```

For the **co**-semantics (as defined in Def. 8), the encoding uses an additional constraint rule, which filters out scopes that do not contain all acceptable assignments:

```
:- domain(A, U), acceptable(A, U), not scope(A, U).
```

The ASP program for the **gr**-semantics (as defined in Def. 9) further includes a minimization statement, as shown next:

```
grounded(N):- N = #count {(A,U): scope(A, U)}.
#minimize {N: grounded(N)}.
```

The first rule computes the cardinality of each **co**-scope, while the second optimisation rule keeps the set minimal models only. Note that this cardinality-based checking is sufficient, given that grounded scopes are unique, and thus any non-grounded complete scope will have strictly more argument-element pairs than the grounded one.

For the **pr**-semantics (as defined in Def. 10), additionally to the clauses that compute the **ad**-scopes, we use the following optimisation statement, in order to find the maximal with respect to set inclusion models:

```
#preference(p1,superset){scope(A, U)}.
#optimize(p1).
```

It should be noted that for the **pr**-semantics code only, one needs to use Asprin, a specialised reasoner for computing qualitative and quantitative optimisations in ASP.

Finally, for the **st**-semantics (as defined in Def. 11), the program starts with the clauses that compute the conflict-free scopes (as described above), and then uses the following two rules to filter out those scopes for which an argument-element pair neither attacks nor is attacked by another pair in the same scope:

```
argumentGetsScopeAttackOn(A, U):- domain(A, U),
        attacks(B, A), scope(B, U).
:- domain(A, U), not scope(A, U),
        not argumentGetsScopeAttackOn(A, U).
```

5.2 A Team-Building Example

The ASP implementation of the different AAFD semantics can be proved more helpful when considered as part of a general system that applies logic programming in order to address specific tasks, rather than when executed as standalone programs. A characteristic case is provided next, where an AAFD encoding can offer leverage both in terms of intuitiveness in modelling declaratively the parameters of a given problem and of implementation easiness. Specifically, inspired by [39], we model a team building example, which is a generalisation of a typical graph colouring problem: given a set of students (characters p to w, in our case) and their friendship relationships, we wish to divide this set into different groups, in such a manner that no student is placed in a group with one or more of her friends. Additionally, we would like the number of such groups to be kept minimal.

The conflict-free semantics of AAFDs is sufficient for producing answers for this problem, but with the incorporation of certain integrity constraints that help eliminate unnecessary results. In particular, the following ASP code listing instantiates the aforementioned problem, starting with the enumeration of students and the specification of

friendship relations among them. We model the latter as attack relations; the idea is to consider each student as an argument, whose scope will be the group she will be assigned to. As such, initially the domain of each student is any available group and the scope that will be generated through the process of calculating AAFD conflict-free extensions will denote the group that each student will be assigned to.

```
% Team Building Example

% Set of students
argument(p;q;r;s;t;u;v;w).

%% Problem instance
% Friendships are denoted as attacks that generate conflicts
attacks(p,q).   attacks(p,r).   attacks(p,v).   attacks(q,p).
attacks(q,r).   attacks(q,t).   attacks(q,u).   attacks(r,p).
attacks(r,q).   attacks(s,t).   attacks(s,u).   attacks(t,s).
attacks(t,q).   attacks(t,u).   attacks(u,q).   attacks(u,w).
attacks(u,t).   attacks(u,w).   attacks(v,p).   attacks(v,w).
attacks(w,v).   attacks(w,u).

% Groups - we specify max at command-line
element(1..max).

% All groups exist in the domain of any node
domain(A, E) :-
    element(E),
    argument(A).

%% Constraint A
% Eliminate cases, where a student belongs to no group
hasNonEmptyScope(A) :-
        argument(A), scope(A, _).

:- argument(X), not hasNonEmptyScope(X).

%% Constraint B
% Eliminate cases, where a student belongs to more than
% a single groups
:- argument(A), scope(A,E1), scope(A, E2), E1!=E2.
```

There are two remarks worth noting about this encoding. First, the generation of all conflict-free extensions produces correct answers, but also generates a number of counter-intuitive answers for the domain extensions. Constraints A and B ensure that we only keep a subset of the extensions that are relevant to the domain, i.e., those where all students belong to exactly one group. As this is a domain-specific requirement, such constraints can be added to complement the more generic AAFD semantics.

Second, initially, one does not know how many groups need to be generated, especially if minimization is required. To keep the encoding simple, we let parameter max (the number of groups to consider) to be treated as a constant provided by the user at command-line (a better solution would be to adopt an incremental solving approach provided by most reasoners, until a solution can be found automatically). This way the user can test different values. For instance, when setting $max = 1$ or $max = 2$, the reasoner cannot compute any satisfiable answer, but when $max = 3$, the reasoner returns 30 equally acceptable distinct solutions, some of which appear next:

```
Answer: 1
scope(p,1) scope(t,1) scope(r,2) scope(u,2)
scope(v,2) scope(q,3) scope(s,3) scope(w,3)
Answer: 2
scope(p,1) scope(t,1) scope(w,1) scope(r,2)
scope(u,2) scope(v,2) scope(q,3) scope(s,3)
...
Answer: 30
scope(q,1) scope(s,1) scope(w,1) scope(r,2)
scope(t,2) scope(v,2) scope(p,3) scope(u,3)
```

The main point that emerges from this example is the easiness of coupling AAFD semantics with constructs offered by logic programming languages, such as integrity constraints in this case, which help approach problems in a succinct and intuitive way. Encoding the same domain using an AAF approach is also feasible, but would require listing a much larger number of relations among arguments, which would cast the solving more complex and the modeling difficult to maintain even for small problem instances.

6 Evaluation

In this section, we conduct an extensive experimental evaluation of our encoding. The goal of this analysis is not to argue that the given implementation is faster or more scalable in comparison to other approaches. Our intention, instead, is to investigate the performance and the breaking points of even a simple implementation of AAFDs, such as ours, while executing tasks ranging from simple to resource-intensive ones. Hopefully, this will enable the knowledge engineer who wishes to model a particular domain with AAFDs to understand what to expect with respect to the domain requirements, and where to focus on for achieving performance gains.

6.1 Data Generation and Experimental Setup

As there are no available datasets containing arguments annotated with their corresponding domains, we decided to generate synthetic data with diverse characteristics, in order to use them as input to the ASP implementation for the calculation of AAFD

extensions. The parameters that vary among the input datasets include the number of arguments ($\#arguments$) and the number of elements of the domain ($\#elements$), the average number of elements assigned to each argument ($\#assignments$), and the average number of attacks made by an argument ($\#attacks$). Specifically, we generated two families (batches) of AAFD datasets:[3]

- the first batch scales with respect to the total number of domain elements: while the $\#arguments$ ranges in $\{2, 3, 4, 5\}$, the $\#elements$ takes values from the set $\{10, 20, 50, 100, 200, 500\}$. In addition, the $\#assignments$ varies between 25%, 50%, 75%, and 100% from a median value randomly chosen from the interval $[\lfloor \frac{N}{2} - X * \frac{N}{2} \rfloor, \lfloor \frac{N}{2} + X * \frac{N}{2} \rfloor]$ (where N is the $\#arguments$ and $X = 0.25, 0.50, 0.75,$ or 1, accordingly). Finally, the $\#attacks$ is a randomly chosen number from the interval $[\lfloor \frac{N}{2} - 0.2 * \frac{N}{2} \rfloor, \lfloor \frac{N}{2} + 0.2 * \frac{N}{2} \rfloor]$. Ultimately, there is a total of 96 cases, for each of which we created 10 different AAFD instances, adding up to 960 different input datasets.
- the second batch scales with respect to the total number of domain arguments: while the $\#elements$ ranges in $\{1, 2, 3, 4\}$, the $\#arguments$ takes values from the set $\{10, 50, 100, 200, 500\}$. The $\#attacks$ varies between $25\%, 50\%, 75\%$, and 100% from a median value randomly chosen from the interval $[\lfloor \frac{N}{2} - X * \frac{N}{2} \rfloor, \lfloor \frac{N}{2} + X * \frac{N}{2} \rfloor]$ (where N is the $\#arguments$ and $X = 0.25, 0.50, 0.75,$ or 1, accordingly). The $\#assignments$ is a randomly chosen number from the interval $[\lfloor \frac{N}{2} - 0.2 * \frac{N}{2} \rfloor, \lfloor \frac{N}{2} + 0.2 * \frac{N}{2} \rfloor]$. Ultimately, there is a total of 80 cases, for each of which we created again 10 different AAFD instances, adding up to 800 different input datasets.

In order to generate these datasets, we developed a randomiser based on *AFBench-Gen2* [19] using Python 3.9 with the module *random*[4]. The randomiser can create an AAFD from scratch with a random number of arguments, elements, attacks and assignments, as requested by the user. The number of attacks and the number of assignments are, however, restricted by the number of arguments and the number of elements of the domain, respectively.

6.2 Experimental Results

Given our dataset of 1760 different AAFDs, we measured the time needed for the reasoner to compute a single (find one/FO$-\sigma$) and all (find all/FA$-\sigma$) scopes for $\sigma \in \{\mathbf{ad}, \mathbf{co}, \mathbf{gr}, \mathbf{pr}, \mathbf{st}\}$. For the FA$-\sigma$ case, we also measured the number of scopes (models) produced. For the $\mathbf{ad}, \mathbf{co}$, and $\mathbf{st}$ semantics, we set a timeout threshold of 15sec, after which we terminated the reasoner, while for the $\mathbf{gr}$ and $\mathbf{pr}$ semantics the timeout threshold was set at 50sec; the decision for applying two different thresholds was driven by the observation that it is much easier for the less restrictive semantics to generate a vast number of models in a very short period of time, in comparison to the ones that require optimisation to be performed before producing results. All experiments were

[3] All input datasets and the corresponding results in CSV format are publicly available and can be downloaded from: https://gitlab.isl.ics.forth.gr/socola/aafwithdomainassignments

[4] https://docs.python.org/3/library/random.html

conducted on a 6-core Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz system, running Ubuntu 20.04.4, Clingo v5.6.2 and Asprin v3.1.1 (Aprin runs with Clingo v5.4.0).

Experiment	AAFD instance 1		AAFD instance 2		AAFD instance 3	
(1st and 2nd batch)	Time	Models	Time	Models	Time	Models
Admissible / 1st batch						
200 elem, 5 arg, 25% var	15.000	11173080+	15.000	10446180+	15.000	14163192+
200 elem, 5 arg, 50% var	15.000	15361037+	15.000	18557308+	15.000	25571634+
200 elem, 5 arg, 75% var	0.022	8192	15.000	9671771+	15.000	20414432+
200 elem, 5 arg, 100% var	15.000	10863923+	15.000	20798364+	15.000	35336935+
500 elem, 5 arg, 25% var	15.000	5031087+	15.000	5977068+	15.000	4917701+
500 elem, 5 arg, 50% var	15.000	3242541+	15.000	4836794+	15.000	7063926+
500 elem, 5 arg, 75% var	15.000	5711497+	15.000	12703934+	15.000	13982966+
500 elem, 5 arg, 100% var	15.000	11965275+	15.000	12504025+	15.000	3802588+
Complete / 1st batch						
200 elem, 5 arg, 25% var	0.027	10368	15.000	12817162+	15.000	16784072+
200 elem, 5 arg, 50% var	15.000	17560949+	0.015	1024	15.000	27262232+
200 elem, 5 arg, 75% var	0.021	2048	15.000	8912602+	15.000	25189504+
200 elem, 5 arg, 100% var	15.000	15271099+	0.013	32	15.000	37449654+
500 elem, 5 arg, 25% var	15.000	5324147+	15.000	7475110+	15.000	6256620+
500 elem, 5 arg, 50% var	15.000	3117130+	15.000	5477005+	15.000	8280617+
500 elem, 5 arg, 75% var	15.000	5868734+	15.000	11660196+	15.000	12432100+
500 elem, 5 arg, 100% var	15.000	10922949+	15.000	11866999+	0.047	4667644+
Admissible / 2nd batch						
4 elem, 200 arg, 25% var	7.548	1	7.596	2	7.278	1
4 elem, 200 arg, 50% var	7.622	1	7.467	4	7.508	1
4 elem, 200 arg, 75% var	7.228	1	7.863	1	7.338	1
4 elem, 200 arg, 100% var	6.227	27	6.439	1	6.944	7
4 elem, 500 arg, 25% var	15	0+	15	0+	15	0+
4 elem, 500 arg, 50% var	15	0+	15	0+	15	0+
4 elem, 500 arg, 75% var	15	0+	15	0+	15	0+
4 elem, 500 arg, 100% var	15	0+	15	0+	15	0+
Complete / 2nd batch						
4 elem, 200 arg, 25% var	12.145	1	12.958	2	10.594	1
4 elem, 200 arg, 50% var	11.070	1	9.742	4	8.159	1
4 elem, 200 arg, 75% var	8.230	1	8.017	1	8.071	1
4 elem, 200 arg, 100% var	6.012	8	7.204	1	7.290	3
4 elem, 500 arg, 25% var	15	0+	15	0+	15	0+
4 elem, 500 arg, 50% var	15	0+	15	0+	15	0+
4 elem, 500 arg, 75% var	15	0+	15	0+	15	0+
4 elem, 500 arg, 100% var	15	0+	15	0+	15	0+

Table 1: A sample of the results recorded for the **ad** and **co** semantics

There are certain general observations that can be made by looking at the results as a whole. For start, there is a distinctive difference in reasoning complexity between the two batches: even the most difficult cases in the first batch (domains with 500 element and 5 arguments) produce a few million of models in a couple of seconds for **ad** and **co** semantics, whereas the corresponding most difficult cases for the second batch (domains with 500 arguments and 4 elements) reach the timeout threshold before producing even a single model. Table 1 presents only a small sample of the experimental results recorded in the CSV files that are available online (a "+" mark next to the num-

ber of models means that the reasoner was forced to stop before exhausting the search space, therefore more models may exist).

Experiment (2nd batch)	Average Time	Mean Time
1 elem, 10 arg, 25% var	.003	.002
1 elem, 10 arg, 50% var	.003	.003
1 elem, 10 arg, 75% var	.003	.002
1 elem, 10 arg, 100% var	.003	.003
1 elem, 50 arg, 25% var	.050	.050
1 elem, 50 arg, 50% var	.049	.049
1 elem, 50 arg, 75% var	.042	.042
1 elem, 50 arg, 100% var	.040	.041
1 elem, 100 arg, 25% var	.292	.293
1 elem, 100 arg, 50% var	.284	.283
1 elem, 100 arg, 75% var	.281	.285
1 elem, 100 arg, 100% var	.253	.251
1 elem, 200 arg, 25% var	2.375	2.359
1 elem, 200 arg, 50% var	2.294	2.334
1 elem, 200 arg, 75% var	2.106	2.101
1 elem, 200 arg, 100% var	2.034	2.090
1 elem, 500 arg, 25% var	15.000	15.000
1 elem, 500 arg, 50% var	15.000	15.000
1 elem, 500 arg, 75% var	15.000	15.000
1 elem, 500 arg, 100% var	15.000	15.000
2 elem, 10 arg, 25% var	.003	.003
2 elem, 10 arg, 50% var	.003	.003
2 elem, 10 arg, 75% var	.004	.004
2 elem, 10 arg, 100% var	.004	.004
2 elem, 50 arg, 25% var	.083	.088
2 elem, 50 arg, 50% var	.070	.068
2 elem, 50 arg, 75% var	.076	.073
2 elem, 50 arg, 100% var	.066	.065
2 elem, 100 arg, 25% var	.523	.520
2 elem, 100 arg, 50% var	.508	.510
2 elem, 100 arg, 75% var	.463	.461
2 elem, 100 arg, 100% var	.494	.490
2 elem, 200 arg, 25% var	4.712	4.633
2 elem, 200 arg, 50% var	4.301	4.342
2 elem, 200 arg, 75% var	4.123	4.100
2 elem, 200 arg, 100% var	3.755	3.811
2 elem, 500 arg, 25% var	15.000	15.000
2 elem, 500 arg, 50% var	15.000	15.000
2 elem, 500 arg, 75% var	15.000	15.000
2 elem, 500 arg, 100% var	15.000	15.000

Experiment (2nd batch, cont.)	Average Time	Mean Time
3 elem, 10 arg, 25% var	.004	.004
3 elem, 10 arg, 50% var	.003	.004
3 elem, 10 arg, 75% var	.003	.003
3 elem, 10 arg, 100% var	.003	.003
3 elem, 50 arg, 25% var	.065	.068
3 elem, 50 arg, 50% var	.054	.054
3 elem, 50 arg, 75% var	.053	.054
3 elem, 50 arg, 100% var	.073	.068
3 elem, 100 arg, 25% var	.542	.539
3 elem, 100 arg, 50% var	.515	.523
3 elem, 100 arg, 75% var	.532	.536
3 elem, 100 arg, 100% var	.466	.470
3 elem, 200 arg, 25% var	4.488	4.464
3 elem, 200 arg, 50% var	4.480	4.513
3 elem, 200 arg, 75% var	4.250	4.293
3 elem, 200 arg, 100% var	4.037	4.104
3 elem, 500 arg, 25% var	15.000	15.000
3 elem, 500 arg, 50% var	15.000	15.000
3 elem, 500 arg, 75% var	15.000	15.000
3 elem, 500 arg, 100% var	15.000	15.000
4 elem, 10 arg, 25% var	.003	.003
4 elem, 10 arg, 50% var	.004	.004
4 elem, 10 arg, 75% var	.004	.003
4 elem, 10 arg, 100% var	.004	.004
4 elem, 50 arg, 25% var	.121	.116
4 elem, 50 arg, 50% var	.114	.112
4 elem, 50 arg, 75% var	.106	.104
4 elem, 50 arg, 100% var	.111	.111
4 elem, 100 arg, 25% var	.843	.844
4 elem, 100 arg, 50% var	.856	.850
4 elem, 100 arg, 75% var	.807	.801
4 elem, 100 arg, 100% var	.822	.835
4 elem, 200 arg, 25% var	7.610	7.572
4 elem, 200 arg, 50% var	7.439	7.490
4 elem, 200 arg, 75% var	7.236	7.283
4 elem, 200 arg, 100% var	6.667	6.862
4 elem, 500 arg, 25% var	15.000	15.000
4 elem, 500 arg, 50% var	15.000	15.000
4 elem, 500 arg, 75% var	15.000	15.000
4 elem, 500 arg, 100% var	15.000	15.000

Table 2: Average and mean execution time of the 2^{nd} batch, for the $\mathbf{ad}$ semantics

To understand if it is the number of arguments or the number of attacks per argument that plays the most important role in making the second batch more computationally intensive, we can look at the average and mean execution times in Table 2: all cases, apart from the ones with 500 arguments terminate before the threshold; moreover, increasing the mean variance of attacks only slightly decreases execution times. This leads to the conclusion that we would probably prefer -at least slightly- to model domains with fewer arguments, even if there are a lot of attacks among them, rather than having big argumentation graphs.

It should be noted that the timeout threshold is also reached in many cases of the first batch too, but this is due to the vast number of models produced there; indeed, another general observation is the much larger set of models produced when having more elements (first batch) in comparison to having more arguments (second batch).

Furthermore, we can also notice in Table 2 that domains with 100 arguments and only a few elements produce all scopes in less than a second. Considering also the very small execution times of the first batch, according to which the number of domain elements weigh much less in comparison to the number of arguments, one can conclude that it would be feasible to model domains with a few hundreds of arguments and elements, even following a simple encoding as ours, at least as far as the $\mathbf{ad}, \mathbf{co}$, and $\mathbf{st}$ semantics are concerned. Of course, increasing the elements would probably increase the number of scopes produced, at least for the less restrictive semantics, which in a realistic setting is not very practical (choosing between a few hundred thousand alternatives may seem arbitrary and would require additional domain heuristics to be applied in order to decide which scope to keep). Still, this finding means that the knowledge engineer can approach a wide range of realistic domains without investing too much effort in applying complex modeling techniques for dealing with performance issues.

Although the same pattern is also observed for the $\mathbf{gr}$ and $\mathbf{pr}$ semantics, available as the supplementary material online, the execution times deteriorate much more steeply: even with a higher threshold, there are many more cases reaching the timeout limit, scattered across all inputs variations, and not just for the extreme cases. The obvious conclusion here is that we would prefer to model domains that produce fewer results under the less restrictive semantics before applying the restrictive ones, so that the calculation of the $\mathbf{gr}$ and $\mathbf{pr}$ semantics would require a much shorter search space to explore and prune, in order to find optimal models[5].

Another finding worth noting is that the gain in execution times observed between the FO$-\sigma$ and the FA$-\sigma$ cases is almost insignificant for all semantics, apart from the $\mathbf{pr}$ ones. This can be attributed to the way grounding is performed with the Clingo reasoner, which needs to be completed before the solving process starts. Incremental grounding and solving is also offered as an option by this reasoner, which, therefore, may be worth investigating for certain domains. The $\mathbf{pr}$ semantics, on the other hand, which are the only ones calculated using the Asprin reasoner, are often more efficiently computed in the FO$-\sigma$ case, especially for the first batch. Asprin is specifically designed for finding optimal models under certain preferences, and although optimisation is also applied for the $\mathbf{gr}$ semantics with Clingo, in that case the reasoner converges to the optimal model, whereas with Asprin optimal models are returned even before solving is completed.

Finally, it may be of interest to point out the high number of unsatisfiable AAFDs, among both batches, for the $\mathbf{st}$ semantics (also available online). With just a few exceptions, their calculation takes less than 2sec to compute, even for the larger datasets.

[5] Note that in the CSVs concerning the $\mathbf{gr}$ cases, the number of models shown correspond to the ones searched for while converging to the single optimal, as returned by Clingo.

7 AAFDs with a background theory

In this section, we consider a specific way to define the domain assignments of an AAFD, based on a logical formalism. This avoids the need to explicitly enumerate the individual elements that are in the domain of an argument, which may be a tedious or time-consuming process.

Our proposal generalises the idea of "background theory", originally proposed in [40]. We argue that this is a more natural and expressive representation of the domains of arguments, which also avoids certain modelling issues. Consider, for instance, the following argument: "all doctors are licensed to practice general medicine". Assume also that it so happens that in a specific context that we want to reason about (e.g., our local hospital), all doctors are female. If we represent the domain of this argument by spelling out the individuals, we lose the information that it is the profession of the individuals that this argument relies on, and not their gender.

The idea is based on the fact that the domain of an argument, viewed from the logical perspective, is essentially a unary predicate. As an example, if we want to say that the domain of an argument a is those hospital doctors that are not surgeons, we could express it using the following rule, which assigns a logical expression to a predicate (say $\delta_a(x)$) that represents the domain of a:

$$r : \delta_a(X) \leftarrow doctor(X), \mathit{not}\ surgeon(X)$$

Extending the idea further, we could consider a logic program Π that encodes the background knowledge (for example information about the roles of the people that work in our local hospital); the domain of a would then contain exactly those elements u of the universe (in our example, people working in our local hospital) that are entailed from r and Π.

More formally, we consider a logic programming language $\mathcal{L}$, as defined in [28]. The signature of a logic program Π in $\mathcal{L}$ consists of a set of constants, a set of variables and a set of predicates. A term is a variable or a constant. An atom is an expression of the form $p(t_1, \ldots, t_n)$, where p is a predicate of arity n and $t_1, \ldots, t_n$ are terms. A literal is an atom a or its negation, $\neg a$. Knowledge is encoded in Π using rules of the form:

$$r : l_1 \leftarrow l_2, \ldots, l_m, \mathit{not}\ l_{m+1}, \ldots, \mathit{not}\ l_n$$

where r is a label, each l_i is a literal in Π, and not denotes default negation. We assume the stable model semantics of logic programs and we use the symbol $\Vdash$ to denote entailment under this semantics.

We define a logic-based AAFD as follows:

Definition 17. *A* logic-based AAFD *is a tuple* $\mathcal{F} = \langle \mathcal{L}, \Pi, \mathcal{A}, \mathcal{R}, \Delta \rangle$ *such that:*

- $\mathcal{L}$ *is a logic programming language that contains (among others) one unary predicate* δ_a *and one unary predicate* ε_a *for each* $a \in \mathcal{A}$
- Π *is a logic program in* $\mathcal{L}^*$*, which is a subset of* $\mathcal{L}$ *that does not contain the* δ_a*,* ε_a *predicates*
- $\mathcal{A}$ *is a set of* arguments

- $\mathcal{R} \subseteq \mathcal{A}^2$ *is a set of* attacks between arguments
- $\Delta = \{r_i^{\Delta} : \delta_a(X) \leftarrow l_1, \ldots, l_m, \textit{not } l_{m+1}, \ldots, \textit{not } l_n \mid i \geq 0, l_j \in \mathcal{L}^*, a \in \mathcal{A}\}$

Intuitively, $\mathcal{L}$ is the language that allows us to express the background knowledge, which is encoded in Π. The constants of $\mathcal{L}$ are essentially the elements of the universe (i.e., the set U in the terminology of Section 3). The domain of each argument a is represented by a unary predicate δ_a, which appears in the head of zero or more rules in Δ and is determined by evaluating the respective rules over Π. In particular, δ_a should be interpreted as follows: an element u is in the domain of an argument a if and only if $\Pi \cup \Delta \Vdash \delta_a(u)$.

The semantics of logic-based AAFDs can be defined in purely logical terms, as shown in the following definitions. We start with the notion of a scope, which, analogously with the domain (Δ), is defined as a set of rules each having as its head a unary predicate ($\varepsilon_a(X)$) specifying the scope of an argument $a \in \mathcal{A}$.

Definition 18. *Consider a logic-based AAFD $\mathcal{F} = \langle \mathcal{L}, \Pi, \mathcal{A}, \mathcal{R}, \Delta \rangle$. A* scope *for $\mathcal{F}$ is a set of rules $E = \{r_i^E : \varepsilon_a(X) \leftarrow l_1, \ldots, l_m, \textit{not } l_{m+1}, \ldots, \textit{not } l_n \mid i \geq 0, l_j \in \mathcal{L}^*, a \in \mathcal{A}\}$*

The intuition behind $\varepsilon_a(u)$ is that u is accepted for a if and only if $\Pi \cup E \Vdash \varepsilon_a(u)$. The semantics are now defined as follows:

Definition 19. *Consider a logic-based AAFD $\mathcal{F} = \langle \mathcal{L}, \Pi, \mathcal{A}, \mathcal{R}, \Delta \rangle$, and a scope E. E is* compliant *if and only if for every $a \in \mathcal{A}$, $u \in U$, $\Pi \cup \Delta \Vdash \delta_a(u)$ whenever $\Pi \cup E \Vdash \varepsilon_a(u)$.*

Definition 20. *Consider a logic-based AAFD $\mathcal{F} = \langle \mathcal{L}, \Pi, \mathcal{A}, \mathcal{R}, \Delta \rangle$, and a scope E. E is* conflict-free *if and only if it is compliant, and for any $(a, b) \in \mathcal{R}$, there is no $u \in U$ s.t. $\Pi \cup E \Vdash \{\varepsilon_a(u), \varepsilon_b(u)\}$.*

Definition 21. *Consider a logic-based AAFD $\mathcal{F} = \langle \mathcal{L}, \Pi, \mathcal{A}, \mathcal{R}, \Delta \rangle$, and a scope E. Consider also some $a \in \mathcal{A}$ and some $u \in U$, such that $\Pi \cup \Delta \Vdash \delta_a(u)$. The pair (a, u) is acceptable with respect to E if and only if, whenever $(b, a) \in \mathcal{R}$ and $\Pi \cup \Delta \Vdash \delta_b(u)$, there exists some $c \in \mathcal{A}$ such that $(c, b) \in \mathcal{R}$ and $\Pi \cup E \Vdash \varepsilon_c(u)$.*

Definition 22. *Consider a logic-based AAFD $\mathcal{F} = \langle \mathcal{L}, \Pi, \mathcal{A}, \mathcal{R}, \Delta \rangle$, and a scope E. E is:*

- Admissible*, if and only if it is conflict-free and for any $a \in \mathcal{A}$, $u \in U$, (a, u) is acceptable with respect to E whenever $\Pi \cup E \Vdash \varepsilon_a(u)$*
- Complete*, if and only if it is admissible and for any $a \in \mathcal{A}$, $u \in U$, $\Pi \cup E \Vdash \varepsilon_a(u)$ whenever (a, u) is acceptable with respect to E*
- Grounded*, if and only if it is complete and, for any complete scope E', $a \in \mathcal{A}$, $u \in U$, $\Pi \cup E' \Vdash \varepsilon'_a(u)$ whenever $\Pi \cup E \Vdash \varepsilon_a(u)$*
- Preferred*, if and only if it is admissible and, for any admissible scope E', $a \in \mathcal{A}$, $u \in U$, $\Pi \cup E \Vdash \varepsilon_a(u)$ whenever $\Pi \cup E' \Vdash \varepsilon'_a(u)$*
- Stable*, if and only if it is conflict-free and, for any $a \in \mathcal{A}$, $u \in U$, whenever $\Pi \cup \Delta \Vdash \delta_a(u)$ and $\Pi \cup E \nVdash \varepsilon_a(u)$, there exists $b \in \mathcal{A}$ such that $(b, a) \in \mathcal{R}$ and $\Pi \cup E \Vdash \varepsilon_b(u)$*

Nixon's diamond can be represented as a logic-based AAFD $\mathcal{F} = \langle \mathcal{L}, \Pi, \mathcal{A}, \mathcal{R}, \Delta \rangle$, with $\mathcal{A} = \{a, b\}$, $\mathcal{R} = \{(a, b), (b, a)\}$,

$$\begin{aligned}
\Delta = \{&\delta_a(X) \leftarrow republican(X), \\
&\delta_b(X) \leftarrow quaker(X)\} \\
\Pi = \{&republican(nixon).\ quaker(nixon). \\
&republican(hoover).\ quaker(hoover). \\
&democrat(hickenlooper).\ quaker(hickenlooper).\}
\end{aligned}$$

Consider the following scopes:

$$\begin{aligned}
E = \{&\varepsilon_a(X) \leftarrow republican(X), \textit{not}\ quaker(X), \\
&\varepsilon_b(X) \leftarrow quaker(X), \textit{not}\ republican(X)\} \\
Scope' = \{&\varepsilon'_a(X) \leftarrow republican(X), \\
&\varepsilon'_b(X) \leftarrow quaker(X), \textit{not}\ republican(X)\} \\
E'' = \{&\varepsilon''_a(X) \leftarrow republican(X), \textit{not}\ quaker(X), \\
&\varepsilon''_b(X) \leftarrow quaker(X)\}
\end{aligned}$$

All three scopes are compliant, conflict-free, admissible and complete. E is also grounded, but neither preferred nor stable. E' and E'' are preferred and stable, but not grounded. According to E, $nixon$ is not in the scope of a ($\Pi \cup E \nvdash \varepsilon_a(nixon)$) or b ($\Pi \cup E \nvdash \varepsilon_b(nixon)$). According to E', he is in the scope of a ($\Pi \cup E' \vdash \varepsilon'_a(nixon)$) but not in the scope of b ($\Pi \cup E' \nvdash \varepsilon'_b(nixon)$). According to E'', he is in the scope of b ($\Pi \cup E'' \vdash \varepsilon''_b(nixon)$) but not of a ($\Pi \cup E'' \nvdash \varepsilon''_a(nixon)$). The same results hold for $hoover$, while $hickenlooper$ is in the scope of b and not in the scope for a according to any of E, E' and E''.

The semantics of logic-based AAFDs are equivalent with the set-based semantics of AAFDs (as defined in Section 3). Specifically, for any AAFD $\mathcal{F} = \langle \mathcal{L}, \Pi, \mathcal{A}, \mathcal{R}, \Delta \rangle$, we can create its corresponding $\mathcal{F}^* = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$, by setting

$$\overline{D^U}(a) = \{u \in U \mid \Pi \cup \Delta \vdash \delta_a(u)\}$$

for all $a \in \mathcal{A}, u \in U$. Then, for any of the semantics σ of AAFDs, it holds that E is a σ-scope of the logic-based AAFD, $\mathcal{F}$, if and only if D^U, defined as follows, is a σ-scope of the set-based AAFD, $\mathcal{F}^*$:

$$D^U(a) = \{u \in U \mid \Pi \cup E \vdash \varepsilon_a(u)\}$$

Moreover, the ASP implementation we presented in Section 5 supports reasoning with logic-based AAFDs. The only difference is that in logic-based AAFDs the input must be in a format like the following, which implements the Nixon's diamond example:

```
argument(a).
argument(b).
domain(a, X):- quaker(X).
domain(b, X):- republican(X).
republican(nixon).
quaker(nixon).
republican(hoover).
quaker(hoover).
democrat(hickenlooper).
quaker(hickenlooper).
```

8 Conclusion

In [40], we introduced a novel argumentation framework, called Abstract Argumentation Framework with Domain Assignments (in short AAFD), where each argument has a domain of application that describes the entities that it refers to and can be applied on. Attacks in an AAFD essentially restrict an argument's domain of application. This framework can be seen as a refinement of AAFs that allows making and reasoning with arguments that do not refer to an individual entity or case, but rather to sets of entities or cases. This enables representing various features of non-monotonic reasoning, such as default knowledge and exceptions, and reasoning with them at the argument level.

In this paper, we extend the results we presented in [40] with: a method for partitioning an AAFD according to a partition of its universe, which enables computing the extensions of an AAFD by combining the extensions of its parts; an alternative logic-based characterization of the domains of arguments, which enables injecting background knowledge into an AAFD; an ASP implementation that computes the scopes of an AAFD and its evaluation; and an example application.

In the future, we plan to extend our theoretical and practical investigations of AAFDs. In terms of theory, we plan to study the relations between AAFDs and other abstract frameworks such as Value-based Argumentation Frameworks [8] and Incomplete Abstract Argumentation Frameworks [7]. In particular, we want to study whether AAFDs can play the role of a unifying framework incorporating the features of other argumentation frameworks. We also plan to integrate the notion of argument domains to other argumentative frameworks, such as Abstract Dialectical Frameworks [15] and Probabilistic Argumentation Frameworks [31]. Another plan is to investigate the relation of AAFDs with structured argumentation frameworks [9], specifically deductive argumentation [10], assumption-based argumentation (ABA) [13], defeasible logic programming (DeLP) [26] and rule-based argumentation (ASPIC$^+$) [37]. By mapping structured arguments with claims that contain variables to abstract arguments with domains, an AAFD will enable reasoning with such arguments, which is not currently possible with AAFs. In more practical terms, we plan to develop more efficient implementations that will exploit features of AAFDs such as equivalence classes of elements and predicates allowing to reason with sets of elements instead of individual elements. We also plan to investigate further applications of AAFDs in domains that involve reasoning with sets of entities or cases, such as negotiation dialogues and legal reasoning.

Acknowledgments

This work has received funding from the Hellenic Foundation for Research and Innovation (H.F.R.I.) and the General Secretariat for Research and Technology (GSRT), under grant agreements No 188 and 4195. We would like to thank Anthony Hunter for his valuable feedback on a previous version of this work.

References

1. Alviano, M.: The pyglaf argumentation reasoner (iccma2021). arXiv preprint arXiv:2109.03162 (2021)
2. Amgoud, L., Ben-Naim, J., Doder, D., Vesic, S.: Acceptability Semantics for Weighted Argumentation Frameworks. In: Proceedings of the 26^{th} International Joint Conference on Artificial Intelligence (IJCAI-17). p. 56–62. AAAI Press (2017)
3. Amgoud, L., Vesic, S.: Rich preference-based argumentation frameworks. International Journal of Approximate Reasoning **55**(2), 585 – 606 (2014)
4. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. The Knowledge Engineering Review **26**, 365 – 410 (2011)
5. Baroni, P., Cerutti, F., Giacomin, M., Guida, G.: AFRA: Argumentation framework with recursive attacks. International Journal of Approximate Reasoning **52**(1), 19–37 (2011)
6. Baroni, P., Rago, A., Toni, F.: From fine-grained properties to broad principles for gradual argumentation: A principled spectrum. International Journal of Approximate Reasoning (IJAR) **105**, 252–286 (2019)
7. Baumeister, D., Neugebauer, D., Rothe, J., Schadrack, H.: Verification in incomplete argumentation frameworks. Artificial Intelligence **264**, 1–26 (2018)
8. Bench-Capon, T.J.M.: Persuasion in Practical Argument Using Value-based Argumentation Frameworks. Journal of Logic and Computation **13**(3), 429–448 (06 2003)
9. Besnard, P., Garcia, A., Hunter, A., Modgil, S., Prakken, H., Simari, G., Toni, F.: Introduction to structured argumentation. Argument & Computation **5**(1), 1–4 (2014). https://doi.org/10.1080/19462166.2013.869764
10. Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artificial Intelligence **128**(1-2), 203–235 (2001)
11. Besnard, P., Hunter, A.: Elements of Argumentation. MIT Press (2008), https://mitpress.mit.edu/books/elements-argumentation
12. Bistarelli, S., Santini, F.: Conarg: A constraint-based computational framework for argumentation systems. In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence. pp. 605–612. IEEE (2011)
13. Bondarenko, A., Toni, F., Kowalski, R.A.: An assumption-based framework for non-monotonic reasoning. In: Proceedings of the 2nd International Workshop on Logic Programming and Non-Monotonic Reasoning. p. 171–189 (1993)
14. Brewka, G., Delgrande, J.P., Romero, J., Schaub, T.: asprin: Customizing answer set preferences without a headache. In: AAAI. pp. 1467–1474. AAAI Press (2015)
15. Brewka, G., Woltran, S.: Abstract dialectical frameworks. In: Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (2010)
16. Budán, M.C., Cobo, M.L., Martinez, D.C., Simari, G.R.: Proximity semantics for topic-based abstract argumentation. Information Sciences **508**, 135 – 153 (2020). https://doi.org/https://doi.org/10.1016/j.ins.2019.08.037
17. Cayrol, C., Lagasquie-Schiex, M.: Bipolar abstract argumentation systems. In: Simari, G.R., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 65–84. Springer (2009)

18. Cerutti, F., Gaggl, S.A., Thimm, M., Wallner, J.P.: Foundations of Implementations for Formal Argumentation. FLAP **4**(8) (2017)
19. Cerutti, F., Vallati, M., Giacomin, M.: A generator for random argumentation frameworks. The 5th International Conference on Control, Mechatronics and Automation (ICCMA) (2017)
20. Dung, P.M.: An argumentation-theoretic foundation for logic programming. The Journal of logic programming **22**(2), 151–177 (1995)
21. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77**(2), 321–357 (September 1995)
22. Dung, P.M., Son, T.C.: Nonmonotonic inheritance, argumentation and logic programming. In: Marek, V.W., Nerode, A., Truszczyński, M. (eds.) Logic Programming and Nonmonotonic Reasoning. pp. 316–329. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
23. Dung, P.M., Toni, F., Mancarella, P.: Some design guidelines for practical argumentation systems. In: Baroni, P., Cerutti, F., Giacomin, M., Simari, G.R. (eds.) Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010. Frontiers in Artificial Intelligence and Applications, vol. 216, pp. 183–194. IOS Press (2010)
24. Dunne, P.E., Hunter, A., McBurney, P., Parsons, S., Wooldridge, M.: Weighted argument systems: Basic definitions, algorithms, and complexity results. Artificial Intelligence **175**(2), 457 – 486 (2011)
25. Dvorák, W., Rapberger, A., Wallner, J.P., Woltran, S.: Aspartix-v19 - an answer-set programming based system for abstract argumentation. In: International Symposium on Foundations of Information and Knowledge Systems (2020)
26. Garcia, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. Theory and Practice of Logic Programming **4**(1-2), 95–138 (2004)
27. GEBSER, M., KAMINSKI, R., KAUFMANN, B., SCHAUB, T.: Multi-shot asp solving with clingo. Theory and Practice of Logic Programming **19**(1), 27–82 (2019). https://doi.org/10.1017/S1471068418000054
28. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gener. Comput. **9**(3/4), 365–386 (1991). https://doi.org/10.1007/BF03037169, https://doi.org/10.1007/BF03037169
29. Horty, J.F., Thomason, R.H., Touretzky, D.S.: A skeptical theory of inheritance in nonmonotonic semantic networks. Artificial Intelligence **42**(2), 311 – 348 (1990). https://doi.org/https://doi.org/10.1016/0004-3702(90)90057-7, http://www.sciencedirect.com/science/article/pii/0004370290900577
30. Hunter, A.: Non-monotonic Reasoning in Deductive Argumentation. CoRR **abs/1809.00858** (2018), http://arxiv.org/abs/1809.00858
31. Hunter, A., Thimm, M.: Probabilistic Reasoning with Abstract Argumentation Frameworks. J. Artif. Intell. Res. **59**, 565–611 (2017)
32. Kaci, S., van der Torre, L.W.N.: Preference-based argumentation: Arguments supporting multiple values. Int. J. Approx. Reason. **48**(3), 730–751 (2008). https://doi.org/10.1016/j.ijar.2007.07.005, https://doi.org/10.1016/j.ijar.2007.07.005
33. Lifschitz, V.: Answer set planning. In: Proceedings of the 1999 International Conference on Logic Programming. p. 23–37. Massachusetts Institute of Technology, USA (1999)
34. Martínez, D.C., García, A.J., Simari, G.R.: On Acceptability in Abstract Argumentation Frameworks with an Extended Defeat Relation. In: Proceedings of the 2006 Conference on Computational Models of Argument: Proceedings of COMMA 2006. p. 273–278. IOS Press, NLD (2006)
35. Modgil, S.: Reasoning about preferences in argumentation frameworks. Artificial intelligence **173**(9-10), 901–934 (2009)

36. Niskanen, A., Järvisalo, M.: μ-toksia participating in iccma 2019. The Third International Competition on Computational Models of Argumentation (ICCMA'19) (2019)
37. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument and Computation **1**, 93–124 (2010)
38. Sillince, J.A.: Shifts in focus and scope during argumentation. Journal of Pragmatics **24**(4), 413–431 (1995)
39. Thadani, S., Bagora, S., Sharmac, A.: Applications of graph coloring in various fields. Materials Today: Proceedings **66**, 3498–3501 (2022), 3rd International Conference on "Advancement in Nanoelectronics and Communication Technologies" (ICANCT-2022)
40. Vassiliades, A., Patkos, T., Flouris, G., Bikakis, A., Bassiliades, N., Plexousakis, D.: Abstract argumentation frameworks with domain assignments. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21). pp. 2076–2082. IJCAI: International Joint Conferences on Artificial Intelligence Organization (2021)
41. Zarefsky, D.: Strategic maneuvering in political argumentation. Argumentation **22**(3), 317 (2008)
42. Zubizarreta, M.L.: The lexical encoding of scope relations among arguments. In: Syntax and the Lexicon, pp. 211–258. Brill (1992)

Appendices

A Proofs for Formal Results

Proof of Proposition 1

We make the following observations:

1. $(a^u, b^u) \in \mathcal{R}_*$ if and only if $a, b \in \mathcal{A}, u \in \overline{D^U}(a) \cap \overline{D^U}(b)$ and $(a, b) \in \mathcal{R}$.
2. If $a^u \in \mathcal{A}_*$, then a^u is acceptable with respect to $\Phi_{nat}(D^U)$ if and only if (a, u) is acceptable with respect to D^U.
3. $D^U \sqsubseteq D^{U\prime}$ if and only if $\Phi_{nat}(D^U) \subseteq \Phi_{nat}(D^{U\prime})$

The proof of observation #1 is direct from the definition of the natural mapping.
The proof of observation #2 follows: take some $a^u \in \mathcal{A}_*$. We first note that, since $a \in \mathcal{A}$, $u \in U$ and $u \in \overline{D^U}(a)$ by the definition of the natural mapping. Moreover:
(a, u) is acceptable with respect to $D^U \Leftrightarrow$ (from Definition 6)
for all $b \in \mathcal{A}$, if $(b, a) \in \mathcal{R}$ and $u \in \overline{D^U}(b)$, then there exists some $c \in \mathcal{A}$ such that $(c, b) \in \mathcal{R}$ and $u \in D^U(c) \Leftrightarrow$ (from observation #1)
for all $a, b \in \mathcal{A}$, $u \in U$, if $(b^u, a^u) \in \mathcal{R}_*$ then there exists some $c^u \in \mathcal{A}_*$ such that $c^u \in \Phi_{nat}(D^U)$ and $(c^u, b^u) \in \mathcal{R}_* \Leftrightarrow$ (from the definition of acceptability in AAFs)
a^u is acceptable with respect to $\Phi_{nat}(D^U)$. This completes the proof of observation #2.
For observation #3, note that $D^U \sqsubseteq D^{U\prime}$ if and only if $u \in D^U(a)$ implies $u \in D^{U\prime}(a)$, which is true if and only if $a^u \in \Phi_{nat}(D^U)$ implies $a^u \in \Phi_{nat}(D^{U\prime})$, i.e., $\Phi_{nat}(D^U) \subseteq \Phi_{nat}(D^{U\prime})$.
Now take some D^U. We will show the result using the above observations for each different $\sigma \in \{\mathbf{cf}, \mathbf{ad}, \mathbf{co}, \mathbf{gr}, \mathbf{pr}, \mathbf{st}\}$.
Case #1: For $\sigma = \mathbf{cf}$, take some D^U that is a $\mathbf{cf}$-scope. Then it is compliant, and:
$u \notin D^U(a) \cap D^U(b)$ whenever $(a, b) \in \mathcal{R} \Leftrightarrow$
it cannot be the case that $u \in D^U(a) \cap D^U(b)$ and $(a, b) \in \mathcal{R} \Leftrightarrow$
it cannot be the case that $(a^u, b^u) \in \mathcal{R}_*$ and $a^u, b^u \in \Phi_{nat}(D^U) \Leftrightarrow$
$\Phi_{nat}(D^U)$ is $\mathbf{cf}$.
Case #2: For $\sigma = \mathbf{ad}$, take some D^U that is an $\mathbf{ad}$-scope. Then:
D^U is a $\mathbf{cf}$-scope and (a, u) is acceptable with respect to D^U, for all $a \in \mathcal{A}$, $u \in D^U(a)$ $\Leftrightarrow$ (by observation #2)
$\Phi_{nat}(D^U)$ is $\mathbf{cf}$ and a^u is acceptable with respect to $\Phi_{nat}(D^U)$ for all $a^u \in \Phi_{nat}(D^U)$ $\Leftrightarrow$
$\Phi_{nat}(D^U)$ is $\mathbf{ad}$.
Case #3: For $\sigma = \mathbf{co}$, the proof is totally analogous to case #2.
Case #4: For $\sigma = \mathbf{gr}$, the proof is direct from case #3, Definition 9 and observation #3.
Case #5: For $\sigma = \mathbf{pr}$, the proof is direct from case #3, Definition 10 and observation #3.
Case #6: For $\sigma = \mathbf{st}$, take some D^U that is a $\mathbf{st}$-scope. Then:
D^U is a $\mathbf{cf}$-scope, and for all $u \in \overline{D^U}(a) \setminus D^U(a)$, there exists $b \in \mathcal{A}$ such that $(b, a) \in \mathcal{R}$ and $u \in D^U(b) \Leftrightarrow$
$\Phi_{nat}(D^U)$ is $\mathbf{cf}$ and for all $a^u \in \mathcal{A}_* \setminus \Phi_{nat}(D^U)$, there exists $b^u \in \mathcal{A}_*$ such that

$(b^u, a^u) \in \mathcal{R}_* \Leftrightarrow$
$\Phi_{nat}(D^U)$ is $\mathbf{st}$.
This completes the proof. □

Proof of Proposition 2

We denote by $[u]$ the equivalence class that contains u, i.e., $[u] = \{u' \in U \mid u' \sim u\}$. Before proceeding with the main proof, we show a number of simple claims that will be useful in the following:

Claim 1. For any $a \in \mathcal{A}$, $u \in U$, and compliant and canonical scope D^U, the following statements are equivalent:

1. $u \in D^U(a)$
2. $[u] \cap D^U(a) \neq \emptyset$
3. $[u] \subseteq D^U(a)$
4. $a^{[u]} \in \Phi_{[nat]}(D^U)$

Proof of claim 1. Take any $a \in \mathcal{A}$, $u \in U$. Then:
$(1) \Rightarrow (2)$ Obvious by the fact that $u \in [u]$.
$(2) \Rightarrow (3)$ Obvious by the fact that D^U is canonical.
$(3) \Rightarrow (1)$ Obvious by the fact that $u \in [u]$.
$(3) \Rightarrow (4)$ Obvious by the definition of $\Phi_{[nat]}$.
$(4) \Rightarrow (3)$ Obvious by the definition of $\Phi_{[nat]}$.

Claim 2. For any $a, b \in \mathcal{A}$, $u \in U$, the following statements are equivalent:

1. $(a^{[u]}, b^{[u]}) \in \mathcal{R}_*$
2. $(a, b) \in \mathcal{R}$, $u \in \overline{D^U}(a) \cap \overline{D^U}(b)$

Proof of claim 2. We have the following:
$(a^{[u]}, b^{[u]}) \in \mathcal{R}_* \Leftrightarrow$ (by the definition of $\mathcal{R}_*$)
$(a, b) \in \mathcal{R}$ and $a^{[u]}, b^{[u]} \in \mathcal{A}_* \Leftrightarrow$ (by the definition of $\mathcal{A}_*$)
$(a, b) \in \mathcal{R}$ and $[u] \subseteq \overline{D^U}(a)$ and $[u] \subseteq \overline{D^U}(b) \Leftrightarrow$ (by claim 1, for the canonical scope $\overline{D^U}$)
$(a, b) \in \mathcal{R}$ and $u \in \overline{D^U}(a)$ and $u \in \overline{D^U}(b) \Leftrightarrow$ (by combining the last two statements)
$(a, b) \in \mathcal{R}$ and $u \in \overline{D^U}(a) \cap \overline{D^U}(b)$.

Claim 3. For any $a \in \mathcal{A}$, $u \in U$ and compliant and canonical scope D^U, such that $u \in \overline{D^U}(a)$, the following statements are equivalent:

1. (a, u) acceptable with respect to D^U in $\mathcal{F}$
2. $a^{[u]}$ acceptable with respect to $\Phi_{[nat]}(D^U)$ in $\mathcal{F}_*$

Proof of claim 3. We have the following:
$(1) \Rightarrow (2)$ Suppose that (a, u) is acceptable with respect to D^U in $\mathcal{F}$. We will show that $a^{[u]}$ acceptable with respect to $\Phi_{[nat]}(D^U)$ in $\mathcal{F}_*$. Indeed, suppose that for an argument

$b^{[u]} \in \mathcal{A}_*$ it is the case that $(b^{[u]}, a^{[u]}) \in \mathcal{R}_*$. Then, by claim 2, $u \in \overline{D^U}(a)$, $u \in \overline{D^U}(b)$ and $(b, a) \in \mathcal{R}$. Since (a, u) is acceptable with respect to D^U in $\mathcal{F}$, it follows that there exists $c \in \mathcal{A}$ such that $(c, b) \in \mathcal{R}$ and $u \in D^U(c)$. Since $u \in D^U(c)$, and D^U is compliant, it follows that $u \in \overline{D^U}(c)$, so $c^{[u]} \in \mathcal{A}_*$, thus (since also $(c, b) \in \mathcal{R}$ and $b^{[u]} \in \mathcal{A}_*$) it holds that $(c^{[u]}, b^{[u]}) \in \mathcal{R}_*$. Moreover, by claim 1, $u \in D^U(c)$ implies that $c^{[u]} \in \Phi_{[nat]}(D^U)$, which shows that $a^{[u]}$ is acceptable with respect to $\Phi_{[nat]}(D^U)$ in $\mathcal{F}_*$.
$(2) \Rightarrow (1)$ Suppose that $a^{[u]}$ is acceptable with respect to $\Phi_{[nat]}(D^U)$ in $\mathcal{F}_*$. We will show that (a, u) is acceptable with respect to D^U in $\mathcal{F}$. Indeed, suppose that for an argument $b \in \mathcal{A}$ it is the case that $(b, a) \in \mathcal{R}$ and $u \in \overline{D^U}(b)$. Then, $a^{[u]}, b^{[u]} \in \mathcal{A}_*$, so $(b^{[u]}, a^{[u]}) \in \mathcal{R}_*$. Since $a^{[u]}$ is acceptable with respect to $\Phi_{[nat]}(D^U)$ in $\mathcal{F}_*$, it follows that there exists some $c^{[u]} \in \mathcal{A}_*$ such that $(c^{[u]}, b^{[u]}) \in \mathcal{R}_*$ and $c^{[u]} \in \Phi_{[nat]}(D^U)$. This implies that for $c \in \mathcal{A}$, $(c, b) \in \mathcal{R}$ and $u \in D^U(c)$, which shows that (a, u) is acceptable with respect to D^U in $\mathcal{F}$.

Claim 4. For any two compliant and canonical scopes D_1^U, D_2^U, the following statements are equivalent:

1. $D_1^U \sqsubseteq D_2^U$
2. $\Phi_{[nat]}(D_1^U) \subseteq \Phi_{[nat]}(D_2^U)$

Proof of claim 4. By the definition of $\Phi_{[nat]}$ and claim 1, we observe that $\Phi_{[nat]}(D_1^U) = \{a^{[u]} \mid a \in \mathcal{A}, u \in D_1^U(a)\}$ (analogously for $\Phi_{[nat]}(D_2^U)$. The result now follows trivially from the above.

Proof of the proposition. Now we can proceed to the proof of the actual proposition, for the different possible values of σ.
For $\sigma = \mathbf{cf}$, we observe the following:
D^U is a $\mathbf{cf}$-scope in $\mathcal{F} \Leftrightarrow$ (by the definition of $\mathbf{cf}$-scope)
there exist no $a, b \in \mathcal{A}$ such that $(a, b) \in \mathcal{R}$ and $u \in D^U(a) \cap D^U(b) \Leftrightarrow$ (by the definition of $\mathcal{R}_*$, the fact that D^U is compliant and canonical, and claim 1)
there exist no $a^{[u]}, b^{[u]} \in \mathcal{A}_*$ such that $(a^{[u]}, b^{[u]}) \in \mathcal{R}_*$ and $a^{[u]}, b^{[u]} \in \Phi_{[nat]}(D^U) \Leftrightarrow$ (by the definition of $\mathbf{cf}$-extension)
$\Phi_{[nat]}(D^U)$ is a $\mathbf{cf}$-extension in $\mathcal{F}_*$.
The case $\sigma = \mathbf{ad}$ follows from the case $\sigma = \mathbf{cf}$, the definition of $\Phi_{[nat]}$, claim 3 and Definition 7.
The case $\sigma = \mathbf{co}$ follows from the case $\sigma = \mathbf{ad}$, the definition of $\Phi_{[nat]}$, claim 3 and Definition 8.
The case $\sigma = \mathbf{gr}$ follows from the case $\sigma = \mathbf{co}$, the definition of $\Phi_{[nat]}$, claim 4 and Definition 9.
The case $\sigma = \mathbf{pr}$ follows from the case $\sigma = \mathbf{ad}$, the definition of $\Phi_{[nat]}$, claim 4 and Definition 10.
For $\sigma = \mathbf{st}$, we observe the following:
D^U is a $\mathbf{st}$-scope in $\mathcal{F} \Leftrightarrow$ (by the definition of $\mathbf{st}$-scope)
for any $a \in \mathcal{A}$, $u \in \overline{D^U}(a) \setminus D^U(a)$, there exists $b \in \mathcal{A}$ such that $(b, a) \in \mathcal{R}$ and $u \in D^U(b) \Leftrightarrow$ (by claim 1, the definition of $\Phi_{[nat]}$, and the fact that D^U is compliant

and canonical)
for any $a^{[u]} \in \mathcal{A}_* \setminus \Phi_{[nat]}(D^U)$, there exists $b^{[u]} \in \Phi_{[nat]}(D^U)$ such that $(b^{[u]}, a^{[u]}) \in \mathcal{R}_* \Leftrightarrow$ (by the definition of $\mathbf{st}$-scope)
$\Phi_{[nat]}(D^U)$ is a $\mathbf{st}$-scope in $\mathcal{F}_*$.
This concludes the proof. □

Proof of Proposition 3

For the first result, we observe that $|\mathcal{F}| = |\mathcal{A}| + |\mathcal{R}| + |\overline{D^U}|$. Moreover, $\mathcal{R} \subseteq \mathcal{A}^2$, whereas the function $\overline{D^U}$ can be modelled as a list of argument-element pairs, each pair appearing at most once in the list, so for its size the following holds: $|\overline{D^U}| \leq |\mathcal{A}| \cdot |U|$. Combining the above observations, we get: $|\mathcal{F}| \leq |\mathcal{A}| + |\mathcal{A}|^2 + |\mathcal{A}| \cdot |U|$.
The second result is direct from the first.
For the third result, we observe that equivalence classes must be disjoint from each other, and their union should be the entire universe U. Thus, $|\mathcal{C}| \leq |U|$. Moreover, suppose that $|\mathcal{A}| = k$ and $\mathcal{A} = \{a_1, \ldots, a_k\}$. Set $S_i = \overline{D^U}(a_i)$ for $i = 1, \ldots, k$. Now take two elements $u_1, u_2 \in U$ such that $u_1 \sim u_2$. It is clear, by Definition 3, that, for all $i = 1, \ldots, k$, $u_1 \in S_i$ if and only if $u_2 \in S_i$. Therefore, each equivalence class is completely characterised by the particular combination of membership relations of its members to each of S_i. This implies that there are at most 2^k equivalence classes. Combining the above, the result follows. □

Proof of Proposition 4

Let us suppose that $\mathcal{F}_1 = \langle \mathcal{A}_1, \mathcal{R}_1 \rangle$, $\mathcal{F}_2 = \langle \mathcal{A}_2, \mathcal{R}_2 \rangle$.
For result #1, note that, by the definition of Φ_{nat}, $|\mathcal{A}_1| \leq |\mathcal{A}| \cdot |U|$. Obviously, $|\mathcal{F}_1| \leq |\mathcal{A}_1| + |\mathcal{A}_1|^2$. Combining the above observations, we get the result.
Result #2 follows directly from result #1 and the fact that $|\mathcal{F}| \geq |\overline{D^U}| = |\mathcal{A}| \cdot |U|$.
For results #3, #4, we observe, by the definition of $\Phi_{[nat]}$, that $|\mathcal{A}_2| \leq |\mathcal{A}| \cdot |\mathcal{C}|$, where $\mathcal{C}$ is the set of equivalence classes of $\mathcal{F}$, under the equivalence relation $\sim$. By Proposition 3, $|\mathcal{C}| \leq \min\{2^{|\mathcal{A}|}, |U|\}$. Moreover, $|\mathcal{F}_2| = |\mathcal{A}_2| + |\mathcal{R}_2| \leq |\mathcal{A}_2| + |\mathcal{A}_2|^2 \leq |\mathcal{A}| \cdot |\mathcal{C}| + |\mathcal{A}|^2 \cdot |\mathcal{C}|^2$. Applying the bound of Proposition 3 on the size of $\mathcal{C}$, we can easily show results #3, #4.
Result #5 follows directly from results #3, #4 and the fact that $|\mathcal{F}| \geq |\overline{D^U}| = |\mathcal{A}| \cdot |U|$.
Result #6 is trivial once we observe that, by construction, $|\mathcal{A}_1| \leq |\mathcal{A}_2|$ and $|\mathcal{R}_1| \leq |\mathcal{R}_2|$.
□

Proof of Proposition 5

Take any reasoning problem in AAFD. To solve it, we can transform the input AAFD (say $\mathcal{F}$) into its respective AAF (say $\mathcal{F}'$) using Φ_{nat}, solve the problem in the AAF realm, and translate the result back into AAFD terminology, exploiting Proposition 1. Let us now compute the complexity of each of those steps.
To start, let us consider an AAFD $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$. Its size is $|\mathcal{F}| = |\mathcal{A}| + |\mathcal{R}| + |\overline{D^U}|$. Set $M = |\mathcal{F}|$.

We set $\mathcal{F}' = \Phi_{nat}(\mathcal{F})$ and $M' = |\mathcal{F}'|$. By Proposition 4 it follows that $M' = O(M^2)$. The transformation process, done using Φ_{nat}, requires $O(M^2)$ time. To see this note the following: for each (a, u) such that $u \in \overline{D^U}(a)$, we create one argument; for each (a, b, u) such that $(a, b) \in \mathcal{R}$ and $u \in \overline{D^U}(a) \cap \overline{D^U}(b)$ we create one attack. The first problem is linear with respect to $|\overline{D^U}|$, and thus linear with respect to M (i.e., $O(M)$). For the second, we need to check, for each $(a, b) \in \mathcal{R}$, whether $u \in \overline{D^U}(a)$ and $u \in \overline{D^U}(b)$. Given that $|\mathcal{R}| \leq M$ and $|\overline{D^U}| \leq M$, this requires quadratic time at worst (i.e., $O(M^2)$).
Finally, the problem of recasting the result into the AAF realm requires time linear with respect to the size of the result, which can be at most equal to the size of the AAF, i.e., $M' = O(M^2)$.
Given the above, and Proposition 1, the result is proved. □

Proof of Proposition 6

We first observe that, for $D^U = \overline{D^U}$, we have that D^U is $\mathbf{cm}$ and $\mathbf{ca}$.
For the other cases, let us consider the AAFD $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}, \overline{D^U} \rangle$ over some universe U. Let us denote by $\mathcal{C}$ the set of all the equivalence classes regarding $\sim$. Set $\mathcal{F}_* = \Phi_{[nat]}(\mathcal{F})$, and suppose that $\mathcal{F}_* = \langle \mathcal{A}_*, \mathcal{R}_* \rangle$. Now take any set $S \subseteq \mathcal{A}_*$, and define the scope D^U_S such that $u \in D^U_S(a)$ if and only if $a^C \in S$ and $u \in C$. It is easy to see that D^U_S is canonical and compliant, and that $\Phi_{[nat]}(D^U_S) = S$.
Now, since $\mathcal{F}_*$ is an AAF, it has at least one $\mathbf{pr}$-scope, say S, thus, by the above results and Proposition 2, D^U_S is a $\mathbf{ca}$-$\mathbf{pr}$-scope in $\mathcal{F}$. Using analogous arguments, we can show the result for $\mathbf{ca}$-$\mathbf{gr}$, $\mathbf{ca}$-$\mathbf{co}$, $\mathbf{ca}$-$\mathbf{ad}$ and $\mathbf{ca}$-$\mathbf{cf}$.
For the case of $\mathbf{st}$-scopes, let us pick, from each equivalence class $C \in \mathcal{C}$, some representative $u_C \in C$ (this is possible, by the Axiom of Choice). For any u, we denote by $\hat{u}$ the representative of u, i.e., $\hat{u} = u_C$, where $u \in C$ (and thus, $u \sim u_C$). Now consider a stable scope D^U. We construct a scope D^U_0, such that, for any $a \in \mathcal{A}$, $u \in U$, $u \in D^U_0(a)$ if and only if $\hat{u} \in D^U(a)$. It is clear that D^U_0 is a $\mathbf{cm}$-scope: indeed, suppose that $u \in D^U_0(a) \setminus \overline{D^U}(a)$. Then, by the definition of D^U_0, the $\sim$ relationship and the fact that $u \sim \hat{u}$, it follows that $\hat{u} \in D^U(a) \setminus \overline{D^U}(a)$, i.e., D^U is not a $\mathbf{cm}$-scope, a contradiction. Moreover, D^U_0 is a $\mathbf{ca}$-scope: indeed, for $u_1 \sim u_2$ it follows that $\hat{u}_1 = \hat{u}_2$ so, for any $a \in \mathcal{A}$, $u_1 \in D^U_0(a)$ if and only if $u_2 \in D^U_0(a)$. Further, D^U_0 is a $\mathbf{cf}$-scope: indeed, it is compliant, so suppose that there exist some $a, b \in \mathcal{A}$, $u \in U$ such that $(a, b) \in \mathcal{R}$ and $u \in D^U_0(a) \cap D^U_0(b)$. Then, $\hat{u} \in D^U(a) \cap D^U(b)$, i.e., D^U is not $\mathbf{cf}$, a contradiction. Finally, we show that D^U_0 is a $\mathbf{st}$-scope: it is $\mathbf{cf}$, so consider some $a \in \mathcal{A}$, $u \in U$ such that $u \in \overline{D^U}(a) \setminus D^U_0(a)$. Then, by construction, $\hat{u} \in \overline{D^U}(a) \setminus D^U(a)$, so, since D^U is $\mathbf{st}$, there exists some $b \in \mathcal{A}$ such that $(b, a) \in \mathcal{R}$ and $\hat{u} \in D^U(b)$, thus $u \in D^U_0(b)$, i.e., D^U_0 is a $\mathbf{st}$-scope. We conclude that D^U_0 is a $\mathbf{ca}$-$\mathbf{st}$-scope, which completes the proof. □

Proof of Proposition 7

By definition, a $\mathbf{cm}$-scope satisfies DC. Also, if D^U is a $\mathbf{cf}$, $\mathbf{ad}$, $\mathbf{co}$, $\mathbf{pr}$, $\mathbf{gr}$, or $\mathbf{st}$ scope, then it is also $\mathbf{cm}$, so it satisfies DC.

For SM, suppose that D^U is a $\mathbf{co}$-scope, and take some $a \in \mathcal{A}$, $u \in \overline{D^U}(a) \setminus D^U(a)$. Suppose, for the sake of contradiction, that there is no $b \in \mathcal{A}$ such that $(b, a) \in \mathcal{R}$ and $u \in \overline{D^U}(b)$. Then, by Definition 6, (a, u) would be acceptable with respect to D^U, thus, since D^U is a $\mathbf{co}$-scope, $u \in D^U(a)$, a contradiction. Thus, D^U satisfies SM. Moreover, if D^U is a $\mathbf{gr}$, $\mathbf{pr}$ or $\mathbf{st}$-scope, then it is also a $\mathbf{co}$-scope, so it satisfies SM.
For SC, it follows by definition that a $\mathbf{cf}$-scope satisfies SC. Also, if D^U is an $\mathbf{ad}$, $\mathbf{co}$, $\mathbf{pr}$, $\mathbf{gr}$, or $\mathbf{st}$ scope, then it is also $\mathbf{cf}$, so it satisfies SC.
For ET, the result follows from Definitions 3, 12, and the formulation of the ET constraint. □

Proof of Proposition 8

Set $D^U = \sqcup_i D^{V_i}$. Before proceeding with the main proof, we show a number of simple claims that will be useful in the following:

Claim 1. For all $\alpha \in \mathcal{A}$, $D^{V_i}(\alpha) = D^U(\alpha) \cap V_i$.
Proof of claim 1. ($\Rightarrow$) If $u \in D^{V_i}(\alpha)$, then $u \in V_i$ by the definition of D^{V_i}, and also $u \in D^U(\alpha)$ by the definition of D^U.
($\Leftarrow$) Take some $u \in V_i$ and $u \in D^U(\alpha)$. By the definition of D^U, there exists some j such that $u \in D^{V_j}(\alpha)$. Since $D^{V_j}(\alpha) \subseteq V_j$ for all j (by the definition of $\mathcal{F}_{V_j}$), and since $V_i \cap V_j = \emptyset$ for all $j \neq i$, it follows that $j = i$, i.e., $u \in D^{V_i}(\alpha)$.
Claim 2. If $u \in D^U(\alpha)$, for some $\alpha \in \mathcal{A}$, then there exists i such that $u \in D^{V_i}(\alpha) \cap V_i$.
Proof of claim 2. Since $u \in D^U(\alpha) \subseteq U$, and $\{\mathcal{F}_{V_i}\}$ is a partition of $\mathcal{F}$, it follows that there exists some i such that $u \in V_i$. Moreover, by Definition 16 and the fact that $D^U = \sqcup_i D^{V_i}$, it follows that there exists some j such that $u \in D^{V_j}(\alpha)$, thus $u \in V_j$. If $i \neq j$, then $V_i \cap V_j = \emptyset$, a contradiction by the fact that $u \in V_i \cap V_j$, so we conclude that $i = j$. This completes the proof of the claim.
Claim 3. If $u \in \overline{D^U}(\alpha)$, for some $\alpha \in \mathcal{A}$, then there exists i such that $u \in \overline{D^{V_i}}(\alpha) \cap V_i$.
Proof of claim 3. Analogous to claim 2.
Claim 4. Take any $\alpha \in \mathcal{A}$, $u \in \overline{D^{V_i}}(\alpha)$ for some i. Then (α, u) is acceptable with respect to D^{V_i} in $\mathcal{F}_{V_i}$ if and only if (α, u) is acceptable with respect to D^U in $\mathcal{F}$.
Proof of claim 4. We first observe that, since $u \in \overline{D^{V_i}}(\alpha)$, it follows that $u \in V_i$, $u \in \overline{D^U}(\alpha)$ and $u \in U$, so the expressions "(α, u) is acceptable with respect to D^{V_i} in $\mathcal{F}_{V_i}$" and "(α, u) is acceptable with respect to D^U in $\mathcal{F}$" are defined. We also note that $\mathcal{R}_{V_i} = \mathcal{R}$. Now we can go ahead and show the claim:
($\Rightarrow$) Take some $(\beta, \alpha) \in \mathcal{R}$, $u \in \overline{D^U}(\beta)$. Then, $(\beta, \alpha) \in \mathcal{R}_{V_i}$. Moreover, since $u \in V_i$, it follows that $u \in \overline{D^U}(\beta) \cap V_i = \overline{D^{V_i}}(\beta)$. Thus, by the hypothesis that (α, u) is acceptable with respect to D^{V_i} in $\mathcal{F}_{V_i}$, it follows that there exists some $c \in \mathcal{A}_{V_i} = \mathcal{A}$ such that $(c, \beta) \in \mathcal{A}_{V_i} = \mathcal{A}$ and $u \in D^{V_i}(c)$. By Claim 1, the latter implies that $u \in D^U(c)$, which completes this part of the proof.
($\Leftarrow$) Take some $(\beta, \alpha) \in \mathcal{R}_{V_i}$, $u \in \overline{D^{V_i}}(\beta)$. Then $u \in V_i$. Moreover, $(\beta, \alpha) \in \mathcal{R}$ and $u \in \overline{D^U}(\beta)$ (by Definition 14). Thus, by the hypothesis that (α, u) is acceptable with respect to D^U in $\mathcal{F}$, it follows that there exists some $c \in \mathcal{A} = \mathcal{A}_{V_i}$ such that $(c, \beta) \in \mathcal{A} = \mathcal{A}_{V_i}$ and $u \in D^U(c)$. By Claim 1, and the fact that $u \in V_i$, the latter implies that $u \in D^{V_i}(c)$, which concludes the proof of the claim.

Now we can proceed to the main part of the proof, for each different σ.

$\sigma = \mathbf{cm}$ ($\Rightarrow$): Take some $\alpha \in \mathcal{A}$, and let $u \in D^U(\alpha)$. Since $u \in D^U(\alpha)$, by Claim 2, there exists some i such that $u \in D^{V_i}(\alpha) \cap V_i$. Since D^{V_i} is a $\mathbf{cm}$-scope, it follows that $u \in \overline{D^{V_i}}(\alpha)$, so $u \in \overline{D^U}(\alpha)$, which shows that $D^U \sqsubseteq \overline{D^U}$, i.e., D^U is a $\mathbf{cm}$-scope.

$\sigma = \mathbf{cm}$ ($\Leftarrow$): Take some $\alpha \in \mathcal{A}$, and let $u \in D^{V_i}(\alpha)$. Since, $u \in D^{V_i}(\alpha)$, by Claim 1, $u \in D^U(\alpha)$. Since D^U is a $\mathbf{cm}$-scope, it follows that $u \in \overline{D^U}(\alpha)$, so by Claim 3, there exists some j such that $u \in \overline{D^{V_j}}(\alpha) \cap V_j$. Since $u \in D^{V_i}(\alpha)$, it follows that $u \in V_i \cap V_j$, but $V_i \cap V_j = \emptyset$ for $i \neq j$, so we are forced to conclude that $i = j$. Thus, $u \in \overline{D^{V_i}}(\alpha)$, which shows that $D^{V_i} \sqsubseteq \overline{D^{V_i}}$, i.e., D^{V_i} is a $\mathbf{cm}$-scope.

$\sigma = \mathbf{cf}$ ($\Rightarrow$): We first note that, since D^{V_i} is a $\mathbf{cf}$-scope for all i, it is also a $\mathbf{cm}$-scope, so, by the previous point, D^U is also a $\mathbf{cm}$-scope. Next, suppose, for the sake of contradiction, that D^U is not a $\mathbf{cf}$-scope. Thus, there exist $\alpha, \beta \in \mathcal{A}$ and $u \in U$, such that $(a, b) \in \mathcal{R}$ and $u \in D^U(\alpha) \cap D^U(\beta)$. By Definition 15, there is a unique i such that $u \in V_i$. Also, since $u \in D^U(\alpha)$, by Claim 2 there exists j such that $u \in D^{V_j}(\alpha) \cap V_j$. But $u \in V_i$ and $V_i \cap V_j = \emptyset$ for $i \neq j$, so we are forced to conclude that $i = j$. Thus $u \in D^{V_i}(\alpha)$. Using analogous arguments, we can show that $u \in D^{V_i}(\beta)$. These conclusions, taken together, contradict the hypothesis that D^{V_i} is a $\mathbf{cf}$-scope, so we are forced to conclude that D^U is a $\mathbf{cf}$-scope.

$\sigma = \mathbf{cf}$ ($\Leftarrow$): We first note that, since D^U is a $\mathbf{cf}$-scope, it is also a $\mathbf{cm}$-scope, so, by the previous point, D^{V_i} is also a $\mathbf{cm}$-scope, for all i. Next, suppose, for the sake of contradiction, that, for some i, D^{V_i} is not a $\mathbf{cf}$-scope. Thus, there exist $\alpha, \beta \in \mathcal{A}_{V_i} = \mathcal{A}$ and $u \in U$, such that $(a, b) \in \mathcal{R}_{V_i} = \mathcal{R}$ and $u \in D^{V_i}(\alpha) \cap D^{V_i}(\beta)$. But then, by Claim 1, $u \in D^U(\alpha) \cap D^U(\beta)$, a contradiction by the fact that D^U is a cf-scope.

$\sigma = \mathbf{ad}$ ($\Leftrightarrow$): The result is direct by Claim 4 and the previous result for $\sigma = \mathbf{cf}$.

$\sigma = \mathbf{co}$ ($\Leftrightarrow$): The result is direct by Claim 4 and the previous result for $\sigma = \mathbf{ad}$.

$\sigma = \mathbf{gr}$ ($\Leftrightarrow$): The result is direct by the previous result for $\sigma = \mathbf{co}$.

$\sigma = \mathbf{pr}$ ($\Leftrightarrow$): The result is direct by the previous result for $\sigma = \mathbf{ad}$.

$\sigma = \mathbf{st}$ ($\Rightarrow$): We first note that, since D^{V_i} is a $\mathbf{st}$-scope for all i, it is also a $\mathbf{cf}$-scope, so, by the previous point for $\sigma = \mathbf{cf}$, D^U is also a $\mathbf{cf}$-scope. Moreover, take some $\alpha \in \mathcal{A}$, $u \in U$ such that $u \in \overline{D^U}(\alpha) \setminus D^U(\alpha)$. By Definition 15, there exists a unique i such that $u \in V_i$. For this particular i, we note that, $u \in \overline{D^U}(\alpha)$ and $u \in V_i$, so by Definition 15, $u \in \overline{D^{V_i}}(\alpha)$. Furthermore, $u \notin D^U(\alpha)$, so $u \notin D^{V_i}(\alpha)$ (by Claim 1). Thus, $u \in \overline{D^{V_i}}(\alpha) \setminus D^{V_i}(\alpha)$, and D^{V_i} is a $\mathbf{st}$-scope by the hypothesis, so there exists some $\beta \in \mathcal{A}_{V_i} = \mathcal{A}$ such that $(\beta, \alpha) \in \mathcal{R}_{V_i} = \mathcal{R}$ and $u \in D^{V_i}(\beta)$. But then, $u \in D^U(\beta)$, which shows that D^U is a $\mathbf{st}$-scope.

$\sigma = \mathbf{st}$ ($\Leftarrow$): We first note that, since D^U is a $\mathbf{st}$-scope, it is also a $\mathbf{cf}$-scope, so, by the previous point for $\sigma = \mathbf{cf}$, D^{V_i} is also a $\mathbf{cf}$-scope for all i. Moreover, take some i, $\alpha \in \mathcal{A}_{V_i}$, $u \in U$ such that $u \in \overline{D^{V_i}}(\alpha) \setminus D^{V_i}(\alpha)$. It follows, by Definition 15, that $u \in \overline{D^U}(\alpha)$ and $u \in V_i$. Also, by Claim 1 and the fact that $u \in V_i$, it follows that $u \notin D^U(\alpha)$, thus $u \in \overline{D^U}(\alpha) \setminus D^U(\alpha)$. But D^U is a $\mathbf{st}$-scope by the hypothesis, so there exists some $\beta \in \mathcal{A} = \mathcal{A}_{V_i}$ such that $(\beta, \alpha) \in \mathcal{R} = \mathcal{R}_{V_i}$ and $u \in D^U(\beta)$. But then, $u \in D^U(\beta)$ and $u \in V_i$, so $u \in D^{V_i}(\beta)$, which shows that D^{V_i} is a $\mathbf{st}$-scope.

For the second part of the proof, we denote by $\sim$ the equivalence relation of indistinguishability for $\mathcal{F}$, and by $\sim_{V_i}$ the respective relation for $\mathcal{F}_{V_i}$. We also note that, since

$\{\mathcal{F}_{V_i}\}$ is the canonical partition, $\{V_i\}$ is the set of equivalence classes for the $\sim$ relation of $\mathcal{F}$. Before proceeding with the proof, we show the following claim:

Claim 5. For any $u_1, u_2 \in U$, the following holds: $u_1 \sim u_2$ if and only if $u_1 \sim_{V_i} u_2$ for all i.

Proof of claim 5. ($\Leftarrow$) Take some $u_1, u_2 \in U$ such that $u_1 \sim u_2$. Then, $u_1, u_2 \in V_i$ for some i, by the construction of $\{V_i\}$. By Claim 1, $\overline{D^{V_i}}(\alpha) = \overline{D^U}(\alpha) \cap V_i$. Combining the latter two with the fact that $u_1 \in \overline{D^U}(\alpha)$ if and only if $u_2 \in \overline{D^U}(\alpha)$ (by the fact that $u_1 \sim u_2$), it follows that $u_1 \in \overline{D^{V_i}}(\alpha)$ if and only if $u_2 \in \overline{D^{V_i}}(\alpha)$. Moreover, for all $j \neq i$, it follows that $u_1, u_2 \notin V_j$, thus $u_1, u_2 \notin \overline{D^{V_j}}(\alpha)$. Combining these two results, we conclude that, if $u_1 \sim u_2$, then $u_1 \sim_{V_i} u_2$ for all i.

($\Rightarrow$) Take some $u_1, u_2 \in U$ such that $u_1 \sim_{V_i} u_2$ for all i. Then, $u_1, u_2 \in V_i$ for some i, by the construction of $\{V_i\}$. By Claim 1, $\overline{D^{V_i}}(\alpha) = \overline{D^U}(\alpha) \cap V_i$. Combining the latter two with the fact that $u_1 \in \overline{D^{V_i}}(\alpha)$ if and only if $u_2 \in \overline{D^{V_i}}(\alpha)$ (by the fact that $u_1 \sim_{V_i} u_2$), it follows that $u_1 \in \overline{D^U}(\alpha)$ if and only if $u_2 \in \overline{D^U}(\alpha)$, i.e., $u_1 \sim u_2$.

Now we can proceed to the main part of the proof:

$\sigma = \mathbf{ca}$ ($\Rightarrow$): Now take some $\alpha \in \mathcal{A}$ and $u_1, u_2 \in U$ such that $u_1 \sim u_2$ and $u_1 \in D^U(\alpha)$. It suffices to show that $u_2 \in D^U(\alpha)$. Indeed, u_1, u_2 belong to the same equivalence class, say V_i, thus $u_1 \in V_i$ and $u_1 \in D^U(\alpha)$, so by Claim 1, $u_1 \in D^{V_i}(\alpha)$. Also, by Claim 5, $u_1 \sim_{V_i} u_2$. Since D^{V_i} is a $\mathbf{ca}$-scope, it follows that $u_2 \in D^{V_i}(\alpha)$, so $u_2 \in D^U(\alpha)$.

$\sigma = \mathbf{ca}$ ($\Leftarrow$): Now take some i, some $\alpha \in \mathcal{A}$ and $u_1, u_2 \in U$ such that $u_1 \sim_{V_i} u_2$ and $u_1 \in D^{V_i}(\alpha)$. It suffices to show that $u_2 \in D^{V_i}(\alpha)$. By Claim 1 $u_1 \in D^{V_i}(\alpha) = D^U(\alpha) \cap V_i$, so $u_1 \in D^U(\alpha)$ and $u_1 \in V_i$. Since $u_1 \sim_{V_i} u_2$ it follows that $u_2 \in V_i$, so $u_1 \sim u_2$. By the hypothesis that D^U is a $\mathbf{ca}$-scope, and the fact that $u_1 \in D^U(\alpha)$ and $u_1 \sim u_2$, it follows that $u_2 \in D^U(\alpha)$. Combining the above, and using Claim 1 again: $u_2 \in D^U(\alpha) \cap V_i = D^{V_i}(\alpha)$, which concludes the proof. $\square$

B ASP Code for the Computation of the AAFD Extensions

The full ASP encoding for computing the FA$-\sigma$ and FO$-\sigma$ problems, for $\sigma \in \{\mathbf{ad}, \mathbf{co}, \mathbf{gr}, \mathbf{pr}, \mathbf{st}\}$, is listed next. In order to execute the code from the command line (and not through the Web interface we also provide, see Section 5), the following commands can be used:

- For $\sigma \in \{\mathbf{ad}, \mathbf{co}, \mathbf{st}\}$:
  $>$ clingo σ.lp someDomain.lp 0
  $>$ clingo σ.lp someDomain.lp 1
- For $\sigma \in \{\mathbf{gr}\}$:
  $>$ clingo σ.lp someDomain.lp –opt-mode=optN
  $>$ clingo σ.lp someDomain.lp –opt-mode=1
- For $\sigma \in \{\mathbf{pr}\}$:
  $>$ asprin σ.lp someDomain.lp 0
  $>$ asprin σ.lp someDomain.lp 1

where the first command in each case computes all valid models, and the second command computes a single model only, if it exists.

```
% Generate all possible assignments of elements to each argument
{scope(A, U) : element(U)} :- argument(A).

%%%%%%%%%%%%%%%%%%%%%%%%%
% Compliant Scope (Def 3)

% Restrict to compliant scopes
:- scope(A, U), not domain(A, U).

%%%%%%%%%%%%%%%%%%%%%%%%%
% Conflict Free Scope (Def 4)

%Import code for compliant scopes

% Restrict to conflict free scopes
:- attacks(A1, A2), scope(A1, U), scope(A2, U).

%%%%%%%%%%%%%%%%%%%%%%%%%
% Admissible Scope (Def 6)

%Import code for conflict free scopes

% Compute acceptability
argumentGetsAttackOn(A, U, B) :-
  domain(A, U), attacks(B, A), domain(B, U).

defendedFrom(A, U, B) :-
    argumentGetsAttackOn(A, U, B), attacks(C, B), scope(C, U).
```

```
notAcceptable(A, U) :-
    argumentGetsAttackOn(A, U, B),
    not defendedFrom(A, U, B).

acceptable(A, U) :-
    domain(A, U),
    not notAcceptable(A, U).

% Restrict to admissible scopes
:- scope(A, U), not acceptable(A, U).

%%%%%%%%%%%%%%%%%%%%%%%%%
% Complete Scope (Def 7)

% Import code for admissible scopes

% Restrict to complete scopes
:- domain(A, U), acceptable(A, U), not scope(A, U).

%%%%%%%%%%%%%%%%%%%%%%%%%
% Grounded Scope (Def 8)
% Import code for complete scopes

% Find the minimal among the complete scopes
grounded(N):- N = #count {(A,U): scope(A, U)}.
#minimize {N: grounded(N)}.
%%%%%%%%%%%%%%%%%%%%%%%%%
% Preferred Scope (Def 9)

% Import code for admissible scopes

% Find maximal among admissible scopes
#preference(p1,superset){scope(A, U)}.
#optimize(p1).

%%%%%%%%%%%%%%%%%%%%%%%%%
% Stable Scope (Def 10)
% Import code for conflict free scopes

% Restrict to stable scopes
argumentGetsScopeAttackOn(A, U):- domain(A, U),
        attacks(B, A), scope(B, U).
:- domain(A, U), not scope(A, U),
        not argumentGetsScopeAttackOn(A, U).
```