A Functional Perspective on Argumentation Schemes

Adam Wyner, Katie Atkinson, and Trevor Bench-Capon

Department of Computer Science, University of Liverpool, Liverpool, UK

Abstract. In multi-agent systems (MAS), abstract argumentation and argumentation schemes are increasingly important. To be useful for MAS, argumentation schemes require a computational approach so that agents can use the components of a scheme to present arguments and counterarguments. This paper proposes a syntactic analysis that integrates argumentation schemes with abstract argumentation. Schemes can be analysed into the roles that propositions play in each scheme and the structure of the associated propositions, yielding a greater understanding of the schemes, a uniform method of analysis, and a systematic means to relate one scheme to another. This analysis of the schemes helps to clarify what is needed to provide denotations of the terms and predicates in a semantic model.

1 Introduction

Argumentation has been useful in multi-agent systems (MAS) to represent dialogue (e.g. [28]), reasoning about action selection (e.g. [22], [1]), and reasoning with inconsistent knowledge bases as in argumentation frameworks [7]. Argumentation schemes (AS) originated in informal logic (e.g. [19, 30]) to represent arguments that are acceptable in ordinary conversation but are classical fallacies. They have also been used in computational argumentation to generate arguments and attacks on arguments, thus integrating ASs into abstract argumentation frameworks [10]. For example, schemes can be viewed as defeasible inference rules in a knowledge base, e.g. as recently in ASPIC+ [6, 20]. A second approach refines the definition of a scheme by providing a semantic model, where the syntactic constituents of the AS have corresponding denotations in the model, e.g. the Practical Reasoning scheme in [5, 4]; an instance of an AS is taken as an argument, and arguments that attack it can be generated with reference to the model. A structural analysis of ASs also appears in [25]. Arguments generated using these approaches can be organised into argumentation frameworks, e.g [10], where arguments and attacks between them can be represented as nodes and arcs in directed graphs. Argumentations frameworks can be evaluated using a variety of semantics, such as grounded, preferred, and stable. A third approach relates ASs in ontological terms [23], which supports the annotation of texts. These approaches are related and address different parts of the computational analysis, use, and evaluation of ASs.

In this paper, we extend the second approach in a novel way by functionally tying together an abstract argument, the role of each proposition in an argumentation scheme, and the structure of the propositions. The result is a formal, fine-grained, and systematic analysis of argumentation schemes that feeds into abstract argumentation. The analysis is presented stepwise, starting from source data and proceeding to successively more abstract levels, along the way providing an intermediate level of representation for

ASs. Instantiated ASs provide arguments suitable for argumentation frameworks, yet have an appropriate internal structure that can be used to capture important elements of the meaning of the original text. We establish a methodology for analysing ASs into their constituent parts that is suitable for computational modelling, in particular, the functional and Logic Programming paradigms. We use our analysis to relate schemes in virtue of common predicates and terms, where the conclusion of one scheme provides part of the justification in some other scheme. This offers a different perspective from the class-subclass relationships that arise naturally when building an ontology of schemes. In our view, it is key to have a detailed analysis of AS syntax in order to provide a basis for development of fine-grained semantic models and for generating attacks from the models. The paper contributes to the theory of MAS argumentation by providing an analysis of ASs that can be used in argumentation between agents.

The chief novel contributions of the paper are: the levels of analysis of ASs from coarse-grained to fine-grained computational forms, the connection between the scheme structure and its constitutive propositions, and the specification of computational relationships between schemes. We can also use the analysis to understand some of the different formulations of ASs as in [27, 12, 20, 4]. From a larger perspective, the analysis begins to bridge the gap between the realisations of argumentation in natural language and their formal analysis. The analysis elaborates [4], where the ASs can be grounded in a semantic model, yielding benefits similar to those in model based diagnosis systems [26, 13], where we can reason from first principles specific to the domain.

In section 2, we discuss the levels of ASs, starting from an instance of an AS and ending with abstract arguments. Initially, the analytic method is illustrated for the Position to Know AS. Section 3 formalises our analysis. The method is extended in section 4 to various arguments about fact, where we also discuss the relationships between schemes. In section 5, we discuss the analysis, related work, and future directions.

2 Levels of Description In Argumentation Schemes

Following [30], an AS as a stereotypical pattern of reasoning in which the premises give a presumptive reason to accept the conclusion. ASs are intended to be used in a dialogical context, where they are given as justifications for a conclusion and are subject to a critique using covering a set of points characteristic of the particular scheme. An interlocutor might pose questions to elicit an answer that either contradicts, reaffirms, or otherwise weakens the rhetorical force of the AS. Consider, for example, the well-known *Argument From Position to Know*, where E is an individual and P a proposition.

– If E is in a position to know P; and
– E asserts P.
– Therefore P.

This is not a logically sound argument, but is widely used and accepted. Like any argument, this argument can be attacked by offering reasons to believe that its conclusion is false (rebuttal), by showing that one of the premises is false (undermining), or by giving a reason to believe that the rule is inapplicable (undercut).[1]

[1] There is not terminological consensus about the ‘parts’ of arguments. For our purposes, we follow the terminology in [20].

To understand, interrelate, implement, and make ASs compatible with argumentation frameworks, we analyse them at different levels of representation. Hitherto, the lack of a formal analysis of the instantiated forms of the schemes as found in informal logic, such as [31], has given rise to confusion, especially in relation to abstract argumentation. This has proven a barrier to the meaningful use of ASs in computational models. To clarify these matters, we introduce a method of analysis that yields several related levels of representation from data to fully abstract arguments, where the scheme level sits in the middle. Table 1 indicates the representation and an indicative reference. Our main, novel proposal is Level 2, which in our view is the appropriate level to refer to an AS that relates to an argumentation framework.

Level	**Representation**	**Reference**
6	Canonical sentences	[31]
5	Labelled roles and strings	[27, 12]
4	Labelled roles and instantiated predicates	[4, 3]
3	Functional roles and instantiated predicates	[20]
2	Functional roles and typed propositional functions	[3]
1	Atomic arguments	[10]

Table 1: Levels of Description of Argumentation Schemes

There are three related objectives of the analysis. First, we want to analyse the characteristic components of a particular AS, giving the roles of the propositions in the AS and the associated internal structure of the propositions. The particular analysis illustrates a method that can be applied to a variety of ASs. Second, we want to relate ASs to arguments that can then be evaluated in abstract argumentation frameworks. Third, we want to relate ASs one to the other. In this section, we address the first two points. We illustrate the method with a particular scheme - Position to Know - and move from a concrete example to successively more analytic representations (from Sixth to First levels in Table 1). Underlying the manner of presentation is the assumption that our analysis proceeds by analysing *fragments*, where a fragment is a subset of the language for which we develop a syntax and semantics [17]; that is, we do not attempt to provide an analysis for all ASs of the "whole" language, which is indeterminate. Each AS uses its own fragment; as we understand additional ASs, we extend the analysis.

What counts as an AS or as different ASs is not clear in the literature, where the same AS can be seen in various formulations, and different catalogues of schemes are given. We presume *canonical* forms, selected from the range of synonymous lexical and syntactic variants, while acknowledging this is subject to further research. Our initial working example is the *Position to Know* AS, where the presumptive conclusion follows from the conjunction of the two premises.

Level 6 - **Canonical sentences**

– Ms. Peters is in a position to know whether Mr. Jones was at the party.
– Ms. Peters asserts that Mr. Jones was at the party.
– Therefore, presumptively, Mr. Jones was at the party.

2.1 From Argumentation Scheme Data to Abstract Arguments

Starting with this AS data in natural language, we incrementally decompose it into its formal, constitutent parts and their relationships, showing the method not only provides the result for this particular scheme, but also shows how to analyse other ASs in a systematic manner. In section 3, we provide our formal language of ASs.

From the data, we identify: the terms and predicates of the argument's propositions, and the roles that propositions play in the AS. We use these to draw out some guiding intuitions. The terms and predicates could be expressed in a suitable logic of the associated expressions and represented in a knowledge base. As for roles, there appear to be two distinct sorts that propositions play in ASs - *scheme generic* and *scheme specific*. The scheme generic roles associate propositions with the *logical role* in the argument irrespective of the propositional content, much as the logical connectives apply to any proposition; in contrast, scheme specific roles highlight aspects of *domain* information in the propositions, where specific schemes present specific content. There is a fixed and finite set of generic roles such as are typically labelled *Premise* and *Conclusion*[2]; in principle, these could apply to any proposition. The scheme specific roles identify the particular role in respect of the propositional content of the proposition that they apply to. For instance, with respect to our sample scheme, we have propositions about an individual being in a position to know, about the assertion that individual makes, and then the conclusion. The scheme requires that we have two premises, one of each type, and linked in a specified way.

These guiding intuitions indicate that it will be useful to represent the specific roles as modifiers of the generic roles, giving us a positionToKnow-Premise, an assertion-Premise, and a conclusion; such labelled roles appear in the appendix of [31], though not consistently nor with further analysis. More specifically, we subsort and label the premises for the role that the sentence plays in the argument. Interestingly, there is no related conception of differentiating the conclusion with respect to specific ASs; this is to do no more than to say that all ASs have a conclusion; we shall have reason to revise this view below.

Level 5 - **Labelled Roles and Strings**

- positionToKnow-Premise: *Ms. Peters is in a position to know whether Mr. Jones was at the party.*
- asserts-Premise: *Ms. Peters asserts that Mr. Jones was at the party.*
- concludes: *Mr. Jones was at the party.*

The assumption here (and the other levels below) is that where the premises conjunctively hold, the conclusion presumptively follows.

[2] There may be alternatives, e.g. *assumption*, *issue*, and *ordinary premise*, as in [20] Definition 3.5, though they are not highly relevant for us at the moment. However, we note the point is a bit subtle. For example, whether a particular proposition is an exception or assumption or ordinary premise (to follow the terminology of [12]) is tied to the propositional content and the relation of the given proposition to the other propositions in the argument; that is, even these generic roles are *contextual* in the argument, and as the context varies, so varies the role. The important point for our purposes is that the generic roles are not themselves labelled with respect to the domain terminology.

At Level 5, there is an implicit relationship between the labels and the content of the strings, which needs to be analytically explicit; after all, the positionToKnow-Premise ought not to label just any string. To draw out this relationship, we represent the predicates and terms, helpfully reusing the role label in the predicate:

Level 4 - **Labelled Roles and Instantiated Predicates**

– positionToKnow-Premise: positionToKnow(*Ms. Peters*, *Mr. Jones was at the party*).
– asserts-Premise: asserts(*Ms. Peters*, *Mr. Jones was at the party*).
– concludes: *Mr. Jones was at the party*.

In other words, we have not only labelled strings, but we have begun to formalise the meaning of the string in a correlated logic-like language.

However, before further analytic steps, we have some additional ingredients to add to our representation. Arguments have a structure *idealised* as *Modus Ponens*, where there are premises that are literals and a rule, from which one infers, given that the literals hold, the conclusion of the rule. There are strict and defeasible rules, symbolically indicated with $\rightarrow$ and $\rightsquigarrow$ respectively. However, rules are rarely explicitly stated as a premise: if someone says *Q since P*, this introduces the presupposition that the speaker has committed to P as well as to a rule *if P, then Q*. As ASs are presumptive, they would include a defeasible rule. We believe that a theory of *circumscription* ([15] and [16]) is useful, wherein we make use of a designated predicate *abCirc* applied to a rule r_i, *abCirc(r_i)*, which means the circumstances are abnormal for the application of the rule r_i. We introduce $\neg$*abCirc(r_i)* as a negated premise (an exception) along with a premise for the rule, which we label *ceteris paribus*, meaning "all other things being equal".[3] Where *abCirc(r_i)* holds, the circumstances are abnormal and *ceteris paribus* is perturbed, so we *cannot* reason to the conclusion even though the rule holds; we have *undercut* the applicability of the rule. Where $\neg$*abCirc(r_i)* holds, the circumstances fulfill *ceteris paribus*, so we *can* reason to the conclusion using the rule. On the other hand, we may have a circumstance which fulfills *ceteris paribus*, yet, it is argued that the rule itself is not correct; here, we have *undermined* the argument by denying the rule premise. While we can give a list of sufficient conditions for abCirc to hold, this list will never, even in principle, be complete. The abCirc predicate and rule appear in our running example as follows, which is still Level 4, but with the auxiliary information about the rule and the abCirc predicate:

Level 4 - **Labelled Roles and Instantiated Predicates with AbCirc and Rule**

– positionToKnow-Premise: positionToKnow(*Ms. Peters*, *Mr. Jones was at the party*).
– asserts-Premise: asserts(*Ms. Peters*, *Mr. Jones was at the party*).
– ceterisParibus-Premise: $\neg$abCirc([positionToKnow(*Ms. Peters*, *Mr. Jones was at the party*) $\wedge$ asserts(*Ms. Peters*, *Mr. Jones was at the party*)] $\rightsquigarrow$ *Mr. Jones was at the party*).
– rule-Premise: [positionToKnow(*Ms. Peters*, *Mr. Jones was at the party*) $\wedge$ asserts(*Ms. Peters*, *Mr. Jones was at the party*)] $\rightsquigarrow$ *Mr. Jones was at the party*.
– concludes: *Mr. Jones was at the party*.

[3] *Ceteris paribus* is used in the physical and social sciences to hold factors constant for the application of the rule.

We can suppose the rule and ceteris paribus premises, so suppress them.

We have a logic-like expression of an ASs. One of our chief goals is to relate ASs to argumentation frameworks, which requires that we associate the scheme with abstract arguments. One approach along these the ASPIC approach to argumentation, best expressed in [20] and referred to as ASPIC+. We will draw comparisons to ASPIC+ in the remainder of this paper, noting where we deviate from it, e.g. *circumscription* deviates from [20] Definition 3.4. We turn to [20] Definition 3.6 and dependent definitions, where arguments are defined syntactically and generic roles are functions from arguments to propositions from the knowledge base. There are several parts to the definition that are relevant to our purpose - arguments, rules, and premises/conclusions. Our analysis proceeds in two analytic steps, first introducing *functions* from arguments to expressions of a Knowledge Base (along the lines of [20]), then introducing propositional functions, which abstract from the propositions of Level 4.

In [20], arguments are *syntactic abstract objects*, $\mathtt{A}$, that are used in an argumentation framework. In contrast, we do not provide a syntactic definition of arguments, but of *ASs*; arguments themselves are *semantic abstract objects* - individuals in the semantic model to which we can apply functions.[4] Furthermore, while arguments may be atomic or complex (having subarguments), we consider for our purposes arguments without subarguments (though we return to this topic later).

Turning to the specific structure of an argument in [20] Definition 3.6, the elements of the arguments (premise, conclusion, and rule) are not just listed informally (as in Levels 4 and 5), but are *functions from arguments*. We assume a Knowledge Base (KB), which in [20] is expressed in some (unspecified) logic. The *Prem* function is from arguments to the set of literals from the KB, giving the premises; the function *Conc* is from arguments to a literal from the KB, giving the conclusion.

In ASPIC+, premises are homogenised rather than differentiated as in Levels 4 and 5. To give more fine-grained semantic information as required for ASs, we subsorted the premise function into several premise functions (and keep a function for the conclusion); thus, the premises are heterogeneous. Furthermore, we want to explicitly associate the premises with the predicates and terms of the correlated expression; to do this, we prefix the premise function with the predicate of the associated expression. Furthermore, as we are considering the *structure* of ASs and not just some particular instance, we want the level of representation where the expressions have not yet been grounded. The justification for subsorting the premises is that literals play different and restricted roles in the AS, and the justification for particular premises of an AS may depend on the particulars of the literal being justified. These distinctions are not made explicit in a homogeneous structure for arguments. For instance, to define the Position to Know AS, there must be *one* proposition that presents the position to know, *one* proposition that presents the assertion, and (aside from the rule and the abCirc predicate) no other premises. Moreover, the justification for the positionToKnow-Premise is different from the justification of the asserts-Premise. Turning to the concludes function, we find that it is important to clearly differentiate conclusions of arguments: the conclusion of an

[4] Arguments have the attributes of abstract objects: they do not “exist” as objects that have physical attributes, yet they can be quantified over (*every argument*), referred to (*that argument*), and bear properties (*an attractive argument*) [2].

AS is semantically particular to that AS, e.g. the meaning of the Position to Know AS conclusion is systematically different from the meaning of the conclusion of the Practical Reasoning AS, and so on for other ASs. Furthermore, where ASs are combined and unified, a point we illustrate in section 4, it is technically important to differentiate conclusions. Therefore, we assume that the conclusion function is differentiated with respect to the AS; in other words, a conclusion of an AS is a conclusion of that particular AS and not of another AS.

Following this analysis, we have functions applied to an argument a_i:

Level 3 - **Functional Roles and Instantiated Predicates**

- positionToKnow-PremiseF(a_i) = positionToKnow(*Ms. Peters*, *Mr. Jones was at the party*).
- asserts-PremiseF(a_i) = asserts(*Ms. Peters*, *Mr. Jones was at the party*).
- ceterisParibus-PremiseF(a_i) = ¬abCirc([positionToKnow(*Ms. Peters*, *Mr. Jones was at the party*). $\wedge$ asserts(*Ms. Peters*, *Mr. Jones was at the party*] $\leadsto$ positionToKnow-ConcludesF(*Mr. Jones was at the party*)).
- rule-PremiseF(a_i) = [positionToKnow(*Ms. Peters*, *Mr. Jones was at the party*) $\wedge$ asserts(*Ms. Peters*, *Mr. Jones was at the party*] $\leadsto$ positionToKnow-ConcludesF(*Mr. Jones was at the party*).
- positionToKnow-ConcludesF(a_i) = *Mr. Jones was at the party*.

However, the previous levels are not, in our view, *schemes* since they are all fully instantiated. To represent a scheme as such, we must abstract over the instantiated expressions. Here we present the formalisation of the working AS; in section 3, we provide a general, formal language of ASs. As predicates are key to our analysis of ASs, we suppose a syntactically typed and semantically sorted language for the expressions of the KB. For example, $\mathrm{X}_{argument}$ is a variable subtyped for arguments, Y_{person} is a variable subtyped with respect to *person*, while $\mathrm{Y}_{literal}$ is a variable subtyped with respect to *literal* (an atomic proposition or its negation).[5] We assume that predicates are also typed and sorted with respect to the types and sorts of their terms (but suppress this in the presentation for readability). The syntactic terms and predicates have denotations in corresponding semantic sorts.[6] We refer to propositions with free variables as *propositional functions* since they are functions from individuals (or ordered n-tuples of individuals) to truth values; the propositional function would, when instantiated, represent a literal or a relation between an individual and a literal.

Turning to the AS functions, we have premise functions associated with predicates, indicated as $\mathcal{X}$*-Premise*$\mathcal{F}$, from variables of type $\mathrm{X}_{argument}$ to the corresponding propositional function, where $\mathcal{X}$ is a string that matches the predicate of the propositional function. We have a concludes function $\mathcal{Y}$*-concludes*$\mathcal{F}$, where the $\mathcal{Y}$ string is the

[5] The variables for literals are a convenience since they too could be analysed as predicates applied to terms, but this requires a more extensive analysis than needed for our purposes.

[6] As we assume only finitely many sorts, we could instead use more verbose expressions in an unsorted first order language with unary predicates that partition the domain of discourse. For instance, rather than Y_{person} and V_{book}, we could instead have predicates such as *person*(x) and *book*(x), where no individual is both a person and book. By the same token, Logic Programming is untyped, but can achieve the effect of typing with special, defined predicates.

label for the AS; in our working AS, *$\mathcal{Y}$-Concludes$\mathcal{F}$* is a function from variables of type X$_{argument}$ to a variable of type Z$_{literal}$. For example, where the premise function is positionToKnow-PremiseF(X$_{argument}$), the output is positionToKnow(Y$_{person}$, Z$_{literal}$); for positionToKnow-ConcludesF(X$_{argument}$), the output is Z$_{literal}$.

Finally, the ceterisParibus-Premise and rule-Premise are constructed with respect to the other premise functions. The ceterisParibus-Premise functions are from variables of type X$_{argument}$ to a expressions of type abCirc, which are propositional functions of the form $\neg$abCirc(rule-PremiseF(X$_{argument}$)); this is well formed when the argument is instantiated. The rule-Premise function is from variables of type X$_{argument}$ to expressions of type Rule. In turn, expressions of type Rule have a specific structure - the body of the rule is the set of premise functions associated with predicates (above), and the head of the rule is a literal from the KB. The formal definitions for both these sorts of premise functions are given in section 3.

We can now represent our working example AS formally in a language compatible with formal argumentation. The important point is that our schematic representation connects the argument with the propositional functions that, given suitable grounding, represent the semantic content of the scheme.

Level 2 - **Functional Roles and Typed Propositional Functions**
- positionToKnow-PremiseF(X$_{argument}$) = positionToKnow(Y$_{person}$, Z$_{literal}$)
- asserts-PremiseF(X$_{argument}$) = asserts(Y$_{person}$, Z$_{literal}$).
- ceterisParibus-PremiseF(X$_{argument}$) = $\neg$abCirc(rule-PremiseF(X$_{argument}$)).
- rule-PremiseF(X$_{argument}$) = [positionToKnow(Y$_{person}$, Z$_{literal}$) $\wedge$ asserts(Y$_{person}$, Z$_{literal}$)] $\rightsquigarrow$ positionToKnow-ConcludesF(X$_{argument}$).
- positionToKnow-ConcludesF(X$_{argument}$) = Z$_{literal}$.

As in Prolog clauses, variables must be consistently instantiated within a scheme. At this level of analysis, we have variables and propositional functions, yet the scheme functions and the predicates are instantiated[7]. In our view, this is the appropriate level to bear the title *argumentation scheme* as such since, properly speaking, a scheme is only a scheme when it has uninstantiated variables for the propositional functions.

Before discussing the next level, we digress. One might propose another abstract scheme where we have an unsorted premise function, which is conceptually equivalent to the premise function as found in ASPIC+ [20] Definition 3.6. Here the output of the function conj-Premise is not a literal or a relation, but a conjunction of literals; the output of conj-Concludes is a literal from among the conjoined literals; the rule is strict, and there is no abCirc predicate. As a whole, the argument has the form of Conjunction Reduction in Propositional Logic; it is unclear to us what such a scheme means where the rule is applicable *ceteris paribus*.

Conjunction Reduction
- conj-PremiseF(a_i^x) = X$_{literal}$, $\wedge$, . . ., $\wedge$ Z$_{literal}$.
- rule-PremiseF(X$_{argument}$) = [X$_{literal}$, $\wedge$, . . ., $\wedge$ Z$_{literal}$] $\rightarrow$ conj-ConcludesF(X$_{argument}$).
- conj-ConcludesF(a_i^x) = Y$_{literal}$.

[7] We would have a second order language with variables for functions and predicates.

While one might take this as a "generic" AS, we propose that it is something different from the AS we find at Level 2 in two respects. First, there are no functions that depend on the semantic content of the literals. Second, it is not an abstract representation of our presumptive AS since conjunction reduction is strict. For these reasons, we prefer to only call structures in the form of Level 2 an AS and not structures of the form for Conjunction Reduction. It may be that there are ASs that have a representation where the premises are unsorted, but this remains an open question for us.

Conjunction Reduction should not be conflated with an AS that has a conjunction of literals. For instance, in an analysis of the Value-based Practical Reasoning AS [4], it is proposed that there are premises that represent the current circumstances and consequences of actions, each of which denote a state that can be specified as a conjunction of literals. As with other ASs, such a scheme requires some extension to the fragment currently under discussion, in particular to express circumstances, consequences, actions, and values. This seems straightfoward, as actions and values type variables, and we can allow circumstance and consequence predicates on a conjunction of literals; we have the associated functions on arguments. Because of this, we can say such a scheme is represented along the lines of Level 2.

Returning from the digression, we have Level 1 where we have argument individuals, $\{\mathrm{a}_0,...,\mathrm{a}_n\}$, that are produced by instantiating ASs. Argument individuals are the objects which can be used (given *attacks* as specified in section 3) for evaluation in an argumentation framework as in [10] and subsequent work. Given indexed argument individuals and the functional definition of argumentation schemes, we can always recover the semantic content that the argument is related to; in this respect, we can identify such an argument with the instantiated AS in which it appears, for the AS may be taken as the internal semantic structure of the argument. Abstract arguments as [10] are not associated with any AS, have no internal structure, and cannot justify any conclusion nor justify attacks since the internal structure of the arguments is not available. Rather abstract arguments are used in the evaluation of sets of arguments. In contrast, justifications are a primary motivation for our analysis of ASs.

To end this section, it is important to emphasise again that the language we have outlined above is an indicative specification for the fragment, not a formal specification on this or all ASs since the range of alternative interpretations or possible expressions remains to be determined. More broadly, it remains to be investigated how expressive the language of arguments must be to accommodate the expressive range of linguistic forms. But, we have already indicated how this might be done for one AS, and we see how this can be done for others in section 4. As well, there may be reason to further analyse the Position to Know AS; for instance, properly speaking, the 'object' of the predicate 'know' is better represented as a proposition in an intensional semantics, yet to provide this analysis opens the door to a fuller formal analysis of natural language syntax and semantics [11], which is relevant, but detracts from the focus in this paper on the formal analysis of ASs. On the other hand, we assume that the premise functions are as fine-grained as their corresponding propositional functions.

3 A Functional Language for Argumentation Schemes

In this section, we present our formal language for ASs. We have a language, specifications for ASs, identity conditions, and a definition of the attack relation. We assume a logical language $\mathcal{L}$, a finite language of syntactically typed and semantically sorted predicates and variables. We have illustrated a fragment of such a language above and leave further specification open-ended and extensible since it depends not just on what schemes we want to represent, but also on further analysis of the schemes we have. We assume classical negation $\neg$.[8] There are expressions of type Rule that are strict $\rightarrow$ or defeasible $\rightsquigarrow$; the bodies are conjoined expressions of $\mathcal{L}$ and heads of propositions of $\mathcal{L}$. As a subset of the variables, we have a type and sort for arguments. We also have a designated predicate *abCirc* that applies to expressions of type Rule. Finally, we have a finite set of AS *labels*, $\mathcal{G}$. With respect to the language, we have component definitions.

Definition 1: AS Functions

- A set of functional premises $\mathcal{D}$, where the functions have the form *$\mathcal{X}$-PremiseF($X_{argument}$)* and are from variables of type $\mathrm{X}_{argument}$ to the propositional function where the predicate is $\mathcal{X}$. The predicates $\mathcal{X}$ are a subset of the predicates of $\mathcal{L}$.
- A set of functional ceterisParibus premises $\mathcal{N}$, where the functions have the form *ceterisParibus-PremiseF($X_{argument}$)* and are from variables of type $\mathrm{X}_{argument}$ to a propositional function of the form $\neg$abCirc($\mathcal{Z}$), and $\mathcal{Z}$ is of type Rule and the rule connective is $\rightsquigarrow$.
- A set of functional rule premises $\mathcal{R}$, where the functions have the form *rule-PremiseF($X_{argument}$)* and are from variables of type $\mathrm{X}_{argument}$ to an expression of type Rule.
- A set of functional conclusions $\mathcal{C}$, where the functions have the form *$\mathcal{Y}$-ConcludesF($X_{argument}$)* and are from variables of type $\mathrm{X}_{argument}$ to a propositional function $\mathcal{Z}$ where the predicate of $\mathcal{Z}$, and $\mathcal{Y}$ is an element of $\mathcal{G}$.

To define ASs, we specify them in terms of their premise and conclusion functions. There are constraints first on the distribution of functions, then constraints between the functions. Expressions such as $\mathcal{N}_{AS_i}$ index functions (or sets of functions) to an AS.

Definition 2: Function Distribution

- There is a non-empty set $\mathcal{D}_{AS_i} \subset \mathcal{D}$; and
- There is exactly one function $\mathcal{N}_{AS_i} \in \mathcal{N}$; and
- There is exactly one function $\mathcal{R}_{AS_i} \in \mathcal{R}$; and
- There is exactly one function $\mathcal{C}_{AS_i} \in \mathcal{C}$.

Definition 3: Function Constraints

- For $\mathcal{N}_{AS_i}$ of form *ceterisParibus-PremiseF($X_{argument}$)*, $\mathrm{X}_{argument} = \mathcal{R}_{AS_i}$; and
- For $\mathcal{R}_{AS_i}$, where the output expression is a rule with body $\mathcal{B}$ and head $\mathcal{H}$, $\mathcal{B} = \mathcal{D}_{AS_i}$ (the set interpreted conjunctively), and $\mathcal{H} \in \mathcal{L}$; and
- $\mathcal{C}_{AS_i} = \mathcal{H}$.

[8] Though see discussion in [20] on a *contrariness function*.

Otherwise, there are no constraints on well-formed ASs. However, this makes explicit generic problems for a theory of ASs. First, just what makes $\mathcal{D}_{AS_i}$ *coherent* since in principle, any random combinations of expressions from $\mathcal{L}$ could form a set $\mathcal{D}_{AS_i}$. Second, just what relationship holds between the body and head of a rule since again in principle, any set $\mathcal{D}_{AS_i}$ by rule imply some random expression in the language. This general issue remains open to further exploration.

Finally, we define an AS as the set of functions meeting the specifications above:

Definition 4: Definition of AS Well-formedness
An AS$_i$ is well-formed if and only if the functions are as defined as in Definitions 1-3, and AS$_i$ = $\mathcal{D}_{AS_i} \cup \{\mathcal{N}_{AS_i}\} \cup \{\mathcal{R}_{AS_i}\} \cup \{\mathcal{C}_{AS_i}\}$.

Set expressions of ASs are useful for relating schemes. Abstractly, the order of the expressions in a scheme matters little; where we represent ASs with sentences, we assume that linguistic dependencies (e.g. pronominal anaphor or sequence of tense) have been resolved.

Of immediate interest is that we can say that two ASs are identical if and only if they are the same set of premise and conclusion functions:

Definition 5: AS Identity Conditions
AS$_i$ and AS$_j$ are identical if and only if AS$_i$ = AS$_j$, otherwise AS$_i \neq$ AS$_j$.

Other relationships between schemes can be defined set-theoretically, but we leave these to future work, turning instead to the *attack* relation.

Attack between arguments in abstract argumentation frameworks [10] is denoted by a relation between two arguments. Where we have instantiated arguments, *attack* is defined with respect to the expressions from the KB that appear in the schemes, e.g. [20] Definitions 3.12-3.16. There are issues concerning the definition of attack bearing on a notions of *contrariness*, though we leave this aside for the purposes of our discussion here, presuming instead that we have classical propositional negation. We express attacks schematically, however, the expressions are only true of instantiated arguments, whereupon the functions produce a truth-bearing expression.

Definition 6: Argument Attack
Argument Attack: X$_{argument}$ *attacks* Y$_{argument}$, X$_{argument} \neq$ Y$_{argument}$, if and only if X$_{argument}$ *rebuts* Y$_{argument}$, or X$_{argument}$ *undermines* Y$_{argument}$, or X$_{argument}$ *undercuts* Y$_{argument}$.

- X$_{argument}$ *rebuts* Y$_{argument}$ if and only if $\mathcal{X}$-ConcludesF(X$_{argument}$) = ¬($\mathcal{Y}$-ConcludesF(Y$_{argument}$)), for $\mathcal{X} \neq \mathcal{Y}$.
- X$_{argument}$ *undermines* Y$_{argument}$ if and only if
  + $\mathcal{X}$-ConcludesF(X$_{argument}$) = ¬($\mathcal{Z}$-PremiseF(Y$_{argument}$)), for some premise prefix $\mathcal{Z}$; or
  + $\mathcal{X}$-ConcludesF(X$_{argument}$) = ¬(rule-PremiseF(Y$_{argument}$)).
- X$_{argument}$ *undercuts* Y$_{argument}$ if and only if $\mathcal{X}$-ConcludesF(X$_{argument}$) = ¬(ceterisParibus-PremiseF(Y$_{argument}$)).

Auxiliary concepts of attack, e.g. defeat relativised to preferences or values, may be defined similarly. It is worth emphasising that attacks are only well-defined where contradictory propositions can be found in the (inconsistent) knowledge base, meaning that there are no 'ad hoc' attacks. At this point, we can turn attention to the structure of a set of closely related schemes.

4 Relating Schemes

Having introduced a method and a formalisation illustrated with a worked example, we consider richer schemes and their relationships. A compendium of ASs is provided in [31]. Assuming that all schemes are represented in (or translated to) our formalisation of ASs, relationships between ASs can be expressed in set theoretic terms of subset and function equivalence with given argument instances. We discuss a family of schemes for arguing abou facts to illustrate these relationships.

In 4.1, we discuss two ways to analyse ASs about facts: first, we take several schemes in their current presentation and express them in our formalisation, showing several limitations; then, because of the limitations, we propose a reanalysis of the schemes into a main scheme with subsidiary ASs for particular premises, yielding a 'tree-like' structure of related ASs (cf. [14]). Among the strengths of our analysis, it helps to clarify fundamental issues about particular analyses of ASs, which can lead to improved ASs, and can computationally relate ASs.

4.1 Arguing about Facts

In [31], several related ASs are reported to establish matters of fact:

1. **Argument from Position to Know (PK)**: Source S is in a position to know about things in subject domain D containing proposition P; and S asserts that P is true (false). Therefore, P is true (false)
2. **Argument from Expert Opinion (EO)**: Source E is an expert in subject domain D containing proposition P; and E asserts that proposition P is true (false). Therefore, P is true (false)
3. **Argument from Witness Testimony (WT)**: Witness W is in position to know whether P is true or not; Witness W is telling the truth (as W knows it); and Witness W states that P is true (false). Therefore, P may be plausibly taken to be true (false)
4. **Argument from Perception (P)**: Person P has a ϕ image (an image of a perceptible property); and to have a ϕ image (an image of a perceptible property) is a *prima facie* reason to believe that the circumstances exemplify ϕ. Therefore, it is reasonable to believe that ϕ is the case.

There appear to be two directions to take the analysis, and we illustrate them each to highlight how our approach shows issues concerning AS analysis into relief. In one way of analysis, the *direct analysis*, we take the ASs essentially as they are, represent their logical forms, and then relate them. In the second approach, *reanalysis*, we reanalyse the variant schemes into essentially a 'theme' and 'variations', the idea being that we a main scheme, *Credible Source*, and variant ways to argue for one of the premises.

Direct Analysis These schemes clearly are related, yet there are a range of variations which are not clearly relevant to the argument, and they should be canonicalised first (though into just what form may be open to debate). For example, all the conclusions are presumptive statements that some state of affairs holds. Yet, this only appears in WT. As well, we assume the arguments are always about reasonable belief. In other schemes there is variation such as *asserts that A*, *asserts that proposition A*, and *states that*. We make these canonical and present them as ASs for those which we have not yet analysed; some complex predicates in the data are decomposed in the analysis. In the following, we have proposed logical forms for the ASs, including Position to Know. Consistent with our assumptions above, the conclusion functions are all labelled according to the AS in which they appear to *relativise* the conclusion to the argument in which it appears.

Position to Know (PK)

- positionToKnow-PremiseF(X$_{argument}$) = positionToKnow(Y$_{person}$, Z$_{proposition}$)
- asserts-PremiseF(X$_{argument}$) = asserts(Y$_{person}$, Z$_{proposition}$)
- positionToKnow-ConcludesF(X$_{argument}$) = Z$_{proposition}$

Expert Opinion (EO)

- expertAbout-PremiseF(X$_{argument}$) = expertAbout(Y$_{expert}$, Z$_{domain}$)
- inDomain-PremiseF(X$_{argument}$) = inDomain(X$_{proposition}$, Z$_{domain}$)
- asserts-PremiseF(X$_{argument}$) = asserts(Y$_{expert}$, X$_{proposition}$)
- expertOpinionAS-ConcludesF(X$_{argument}$) = X$_{proposition}$

Witness Testimony (WT)

- positionToKnow-Premise(X$_{argument}$) = positionToKnow(Y$_{witness}$, Z$_{proposition}$)
- truthTelling-PremiseF(X$_{argument}$) = truthTelling(Y$_{witness}$)
- asserts-PremiseF(X$_{argument}$) = asserts(Y$_{witness}$, X$_{proposition}$)
- witnessTestimony-ConcludesF(X$_{argument}$) = X$_{proposition}$

Perception (P)

- perceives-PremiseF(X$_{argument}$) = perceives(Y$_{person}$, Z$_{percept}$)
- perceptSupports-PremiseF(X$_{argument}$) = perceptSupports(Z$_{percept}$, V$_{proposition}$)
- perception-ConcludesF(X$_{argument}$) = V$_{proposition}$

A range of issues arise about particular analyses, variations in the terms and predicates, and attendant relationships between the schemes. We see that the ASs intersect on some functions and not on others: PK and WT have *positionToKnow*; PK, EO, and WT have *asserts*; only P has *perceptSupports*. The conclusions are all differentiated, which makes sense since a proposition that concludes one proposition is not sufficient to make the conclusion of some other AS hold. However, these are shallow observations.

A deeper level of observations touch on the issue of the fine-grained semantic analysis of the statements of the AS. To give one example, we see that several ASs have the function *asserts*-PremiseF(X$_{argument}$), which given our functional assumptions, ought to map to the same propositional functions. Yet, clearly they do not; in particular, we have predicates such as *asserts* applying to terms of different sorts such as *person*, *expert*, and *witness* . This might be taken to imply that either we have predicates similarly subsorted, to distinguish schemes, or we have polymorphic types. However, articulating the logical forms of the propositions, we would need neither, simplifying the task. Supposing a Neo-Davidsonian Event-theoretic Semantics [18], the predicate relates to its terms indirectly (by events *e* and thematic roles *agent* and *theme*), leaving the subsorting of the terms as an independent element. Simplifying, we have:

$\exists e\ \exists x\ [asserts(e) \wedge agent'(e,x) \wedge theme'(e,P) \wedge expert'(x)]$,
which is the logical form that means *There is an event that is an event of asserting, and there is an individual which is the agent of the asserting event, the theme of the asserting event is a proposition P, and the individual is an expert.*

Logical forms for other propositional functions would vary with respect to *expert′(x)*, introducing *witness′(x)* and *person′(x)* instead.

Nonetheless, this does not quite solve the issue concerning our functional assumption since even with such a logical form, the propositional functions vary. One possible solution would be simply to abstract over the predicate that varies and define its possible scope, e.g. $\mathcal{R} = \{\text{expert}', \text{witness}', \text{person}'\}$, which essentially represents the set of sets of individuals that play a part in such AS; the logical form for the assertion propositional function would have this variable in place of *expert′(x)*. In effect, the particular ASs are tied to a higher level abstract AS that comes with an associated ontology.

While we might treat the asserts-Premise in a relatively homogeneous manner, others are more problematic. On the face of it, there seems to be some semantic relationship between *positionToKnow*, *expertAbout*, and (perhaps) *perceives*. Yet, unless this relationship further specified (as we have done with respect to *asserts*), the ASs are otherwise unrelated. A similar point applies *asserts* and *perceptSupports*.

Unless and until these lexical semantic relationships between terms and predicates are resolved, we cannot say much further about the relationships between the associated ASs. The discussion is instructive in any case, for by formally clarifying the space of issues, we can systematically address them in a coherent manner. For example, one approach is to reanalyse the schemes to homogenise their differences, which are made subsidiary to a main scheme.

Reanalysis In contrast to the direct analysis, we reanalyse the four schemes as subsidiary to or dependent on a single main scheme, the *Argument from Credible Source* (CS), which we articulate below; each of the four dependent schemes is a way of establishing a premise of the main scheme. In other words, the CS along with particular additional arguments for premises form a tree-like structure, where one (subsidiary) argument justifies the premise of another argument. This can be seen as an overall argument with subarguments, where the subarguments are *glued* together by *unification* of variables along with sortal restrictions on terms and predicates. The idea is that we can have a cascade of rules, where more specialised rules justify predicates that can unify with predicates in the body of some more abstract rule. For instance, schemes for eyewitness testimony or testimony from videotape may be used to justify predicates in a more generalised scheme for witness testimony; in turn, witness testimony can be used to justify predicates in yet a more abstract scheme for credible source. An analysis along these lines is compatible with a Logic Programming paradigm: a predicate $\mathcal{P}_i$ in the body of a rule $\mathcal{R}_i$ unifies with the predicate $\mathcal{P}_j$ that is the head of some other rule $\mathcal{R}_j$; the predicate $\mathcal{P}_j$ has to be *justified* (true) with respect to the the rule $\mathcal{R}_j$, and by the same token we can say that $\mathcal{R}_j$ justifies $\mathcal{P}_i$. There may be alternative ways to justify $\mathcal{P}_i$, given different rules $\mathcal{R}_k$,...,$\mathcal{R}_l$ with a predicate in the head that can unify with $\mathcal{P}_i$. As well, there may be rules that justify other predicates that are in the body of the rule $\mathcal{R}_i$. Similar points can be made for justifications of predicates in the body of $\mathcal{R}_j$. In the following, we give a particular instance of such a cascade.

We first formalise the CS, followed by formalisations of the other schemes, and show how we can relate the schemes set theoretically to yield the tree-like structure. We give the CS in two forms - Level 6 and Level 2.

Canonical Sentences

- John is a credible source about the domain of ornithology.
- John says that female blackbirds are brown.
- That female blackbirds are brown is a statement in the domain of ornithology.
- Therefore, presumptively, female blackbirds are brown.

Functional Roles and Typed Propositional Functions

- credibleSourceAbout-PremiseF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)
- assertion-PremiseF(X$_{argument}$) = asserts(Y$_{person}$, Z$_{proposition}$)
- statementInDomain-PremiseF(X$_{argument}$) = statementInDomain(Z$_{proposition}$, V$_{domain}$)
- credibleSource-ConcludesF(X$_{argument}$) = Z$_{proposition}$

Returning to our four other schemes (PK, WT, EO, P), we claim these are different ways of arguing for the premise *credibleSourceAbout*; that is, each of these schemes has as conclusion the propositional content associated with the *credibleSourceAbout* premise of the CS. So, the schemes are different in terms of structure, yet related in terms of content. Our analysis extracts conceptual redundancy from the current analysis of schemes and formally homogenises them. In *Position to Know*, we extract the assertion premise and make the conclusion into the function for credible source:

EO′

- expertAbout-PremiseF(X$_{argument}$) = expertAbout(Y$_{person}$, V$_{domain}$)
- expertOpinion′-ConcludesF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)

PK′

- positionToKnow-PremiseF(X$_{argument}$) = positionToKnow(Y$_{person}$, V$_{domain}$)
- positonToKnow′-ConcludesF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)

WT′

- witnessConcerning-PremiseF(X$_{argument}$) = witnessConcerning(Y$_{person}$, V$_{domain}$)
- witnessTestimony′-ConcludesF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)

P′

- perceivesConcerning-PremiseF(X$_{argument}$) = perceivesConcerning(Y$_{person}$, V$_{domain}$)
- perceives′-ConcludesF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)

Clearly, EO′, PK′, WT′, and P′ all have the same propositional function as a conclusion, so they are all seem to be *about* making the same point, though they differ in how the premises justify the conclusion. More importantly, we can equate the conclusion of one AS (given a value for $\mathcal{Y}$) with the premise of another since the following is true:

credibleSourceAbout-PremiseF(X$_{argument}$) = $\mathcal{Y}$-ConcludesF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)

And in general, the premises and conclusions of ASs may be related as follows:

> **Definition 7: Premise-Conclusion Tie**
> $\mathcal{X}$-PremiseF(X$_{argument}$) = $\mathcal{Y}$-ConcludesF(X$_{argument}$) = V$_{proposition}$,
> where $\mathcal{X}$ and $\mathcal{Y}$ are labels in different ASs.

Strictly speaking, this is not necessary to state in presentation of the relationship of the ASs since unification of variables will take care of this.

Suppose that we are making use of the PK$'$, CS, and (for clarity) the equivalence of the premise and conclusion. Prior to instantiation, we have:

- **PK$'$**
  - positionToKnow-PremiseF(X$_{argument}$) = positionToKnow(Y$_{person}$, V$_{domain}$)
  - positionToKnow$'$-ConcludesF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)
- **CS**
  - credibleSourceAbout-PremiseF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)
  - assertion-PremiseF(X$_{argument}$) = asserts(Y$_{person}$, Z$_{proposition}$)
  - statementInDomain-PremiseF(X$_{argument}$) = statementInDomain(Z$_{proposition}$, V$_{domain}$)
  - credibleSource-ConcludesF(X$_{argument}$) = Z$_{proposition}$
- **Premise-Conclusion Equivalence**
  - credibleSourceAbout-PremiseF(X$_{argument}$) = positionToKnow$'$-ConcludesF(X$_{argument}$) = credibleSourceAbout(Y$_{person}$, V$_{domain}$)

When we instantiate the schemes and unify the variables, the denotations of the schemes are tied together. Suppose that for a person *john$'$*, a domain, *ornithology$'$*, and a proposition *female_blackbirds_are_brown$'$*, we have the following instantiation:

- **PK$'$**
  - positionToKnow-PremiseF(a$_i$) = positionToKnow(john$'$, ornithology$'$)
  - positionToKnow$'$-ConcludesF(a$_i$) = credibleSourceAbout(john$'$, ornithology$'$)
- **CS**
  - credibleSourceAbout-PremiseF(a$_i$) = credibleSourceAbout(john$'$, ornithology$'$)
  - assertion-PremiseF(a$_i$) = asserts(john$'$, female_blackbirds_are_brown$'$)
  - statementInDomain-PremiseF(a$_i$) = statementInDomain(female_blackbirds_are_brown$'$, ornithology$'$)
  - credibleSource-ConcludesF(a$_i$) = female_blackbirds_are_brown$'$
- **Premise-Conclusion Equivalence**
  - credibleSourceAbout-PremiseF(a$_i$) = positionToKnow$'$-ConcludesF(a$_i$) = credibleSourceAbout(john$'$, ornithology$'$)

From this, we see the conclusion from PK$'$ is *female_blackbirds_are_brown$'$*.

The other ASs - EO′, WT′, P′ - could be used. They are mutually compatible where we can unify the variables, giving different justifications for the same conclusions; where we have different unifications, then the resultant ASs may be understood as incompatible (or not, depending on other aspects of the given model). Each of these dependent ASs, where an AS$_i$ has a conclusion that matches the premises of another AS$_j$, where AS$_j$ is a dominant scheme in having the premise that is matched. We could have additional schemes to justify other premises of the CS; as well, each of these dependent schemes might have dependent schemes justifying their premises, and so on, giving a *tree* made up of a dominant scheme with branches (the subordinate AS).

There are a range of lexical semantic issues to address. While our direct analysis may have introduced too fine-grained an analysis, the reconstruction here may have homogenised important distinctions in meaning, creating awkward expressions. For instance, is it correct to *witness a domain* or *perceive a domain*? These are different from the more familiar phrases of *being an expert about a domain* or *being in a position to know about a domain*. This suggests some further intermediate structure is required, where the CS has more abstract predicates that are argued for in more fine-grained ASs.

In our instantiation, we have but one argument individual a$_i$ which appears in two distinct ASs; this arises because of the unification of the argument variable. In [20], arguments have subarguments given the recursive syntactic definition of arguments. We could reproduce this semantically. Suppose that in the model, argument individuals are totally ordered by the *subargument* relation and that we have the following constraint:

Definition 8: Subargument
[$\mathcal{X}$-PremiseF(X$_{argument}$) = $\mathcal{Y}$-ConcludesF(Y$_{argument}$) = V$_{proposition}$] $\rightarrow$
subargument(X$_{argument}$, Y$_{argument}$),
where $\mathcal{X}$ and $\mathcal{Y}$ are labels in different ASs, and X$_{argument}$ $\neq$ Y$_{argument}$.

A subargument arises where the instantiation of the conclusion of one AS unifies with the premise of the instantiation of the a different AS; this is not as in [20], where subarguments are defined recursively such that each branch of an argument tree is a subargument and so is each branch of each branch and so on. To capture a similar idea in our analysis, we assume the *Transitive Closure* of the *subargument* relation. The problem for us is the precise meaning, use, and importance of the recursive definition in ASPIC+, particularly where argument attacks are concerned. For our purposes, we leave investigation of these matters for future research.

5 Discussion and Future Work

In this paper, we have presented a functional language for a computational analysis of ASs that is compatible with argumentation frameworks. We have outlined an extensible methodology, worked through an example, and shown how ASs in our analysis can be systematically related to one another. In this section, we discuss some related work, topics that were not addressed in the presentation, and future work.

There is an abundance of research in ASs. [31] provide a catalogue of ASs at a largely descriptive and unsystematic level. A range of considerations about ASs are developed in [21], but these do not give rise to a formal analysis. Some research remains at Level 5, where strings are annotated with respect to labels [27]. There are computational proposals which do not differentiate premises [25, 20], while others do differentiate premises, but not with respect to the content of the associated propositions, e.g. [12]. ASs have been analysed for legal argumentation. In [29], an approach to the analysis of schemes is outlined and exemplified for legal reasoning. There are formalised ASs with propositional functions for legal case-based reasoning [33, 6], though these are highly specialised. Moreover, the approach to abstracting arguments is underspecified and not tied to a theory of argumentation, e.g. ASPIC+. None of these proposals for legal reasoning propose a more general language for ASs. A different line of work provides an interchange format and associated ontology [24], which allows for differentiated premises; however, this work does not tie the premise subsorts to the propositional content, much less to the natural language statements of the argument; nor does it take into account formal argumentation systems. In [8], the formal problems with the AIF are addressed by interpreting it in terms of ASPIC+, yet ASPIC+ is a theory with homogeneous premises, whereas heterogeneous premises are required. Closest in spirit to our proposal is [4, 3], where a natural language expression of practical reasoning is analysed in terms of predicates and terms with respect to denotations in a semantic model. In that approach, ASs are construed as arguments, though the specific translation is underspecified and a language for ASs is not provided.

In sections 2.1 and 4.1, we pointed to issues about the fine-grainedness of premise functions. Fine-grainedness arises as there are many ways of expressing or modifying a predicate: one may *know indirectly* or *know in one's heart*; one may *assert quietly* or *state* or *assert in court*. Similarly, the terms of the predicates may vary with respect to properties of the person or alternative forms of the proposition. In general, this is related to similar discussions about propositions and propositional attitudes, e.g. *belief*, and may warrant a similar treatment [9]. One way to treat some of these issues is to homogenise terminology under synonyms, e.g. *asserts* in the data is substitutable by *said* or *wrote*, so we can presume these map to the same premise function. The knowledge base may be further enriched for such lexical semantic information. Alternatively, one may simply use a controlled language to constrain the expressivity of the language in a give domain. However, this is a general topic that requires significant further research.

In future work, we look forward to a range of topics. Of a particular interest is the relationship of ASs to semantic models, critiques, and dialogue. Current examples of semantic models are the knowledge bases of [20] and the transition systems of [4, 3]. We have not discussed the *dialogical* aspects of argumentation, nor *critical questions*, which are questions used to critique the presumptive truth of the instantiated AS. These topics are closely related, crucial aspects to a full analysis of ASs and argumentation. Briefly, ASs may be critiqued, using critical questions, in the course of a dialogue, where there are alternative instantiations of propositions of the AS relative to the semantic model; the questions *encode* these alternatives. Models are also used to *generate* instantiated ASs, some of which may be in attack relations (cf. on these topics [4, 3]). In analysing a range of ASs, of special interest is the differentiation of semantic models with respect to the requirements of the scheme; in other words, the different fragments of the language are given appropriate semantic models. In [20], three different knowledge bases are provided, but each is of a homogeneous form. We want to differentiate sorts of rules and justify them accordingly, so that, e.g. rules about causation, definition, observational generalisation, values, assertions, are each given a distinct sort of justification (and related attacks), and the meanings of the justifications (attacks) are grounded in an associated semantic model. Finally, we intend to implement ASs in a database for web-based applications[32] and Functional or Logic Programs that instantiate the ASs, generate arguments, calculate attacks, and determine extensions.

6 Acknowledgments

This work was supported by the FP7-ICT-2009-4 Programme, IMPACT Project, Grant Agreement Number 247228. The views expressed are those of the authors. We thank the reviewers for their helpful comments.

References

1. Leila Amgoud, Caroline Devred, and Marie-Christine Lagasquie-Schiex. A constrained argumentation system for practical reasoning. In *AAMAS (1)*, pages 429–436, 2008.
2. Nicholas Asher. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers, 1993.
3. Katie Atkinson, Trevor Bench-Capon, Dan Cartwright, and Adam Wyner. Semantic models for policy deliberation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Law (ICAIL 2011)*, pages 81–90, Pittsburgh, PA, USA, 2011.
4. Katie Atkinson and Trevor Bench-Capon. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171(10-15):855–874, 2007.
5. Trevor Bench-Capon and Katie Atkinson. Argumentation schemes: From informal logic to computational models. In C. Reed and C. Tindale, editors, *Dialogue and Argumentation: An Examination of Douglas Waltonś Theories of Reasoning and Argument*, pages 103–114. Academic Press, London, 2010.
6. Trevor Bench-Capon and Henry Prakken. Using argument schemes for hypothetical reasoning in law. *Artificial Intelligence and Law*, 18(2):153–174, 2010.
7. Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.
8. Floris Bex, Henry Prakken, and Chris Reed. A formal analysis of the aif in terms of the aspic framework. In *Proceedings of the 2010 conference on Computational Models of Argument: Proceedings of COMMA 2010*, pages 99–110, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.
9. Maxwell Cresswell. *Structured meanings: the semantics of propositional attitudes*. Bradford Books. MIT Press, 1985.
10. Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.
11. L.T.F. Gamut. *Logic, Language, and Meaning: Intensional Logic and Logical Grammar*. University of Chicago Press, 1991.
12. Thomas Gordon, Henry Prakken, and Douglas Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171:875–896, 2007.
13. Nada Lavrač, Ivan Bratko, Igor Mozetič, Boian Čerček, Anton Grad, and Matija Horvat. KARDIO-E: an expert system for electrocardiographic diagnosis of cardiac arrhythmias. *Expert Systems*, 2(1):46–55, 1985.

14. Ronald Loui and Jeff Norman. Rationales and argument moves. *Artificial Intelligence and Law*, 3(3):159–189, 1995.
15. John McCarthy. Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.
16. John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence Journal*, 28:89–116, 1986.
17. Richard Montague. The proper treatment of quantification in ordinary english. In Richard Thomason, editor, *Formal Philosophy: Selected Papers of Richard Montague*, pages 247–270. Yale University Press, 1974.
18. Terence Parsons. *Events in the Semantics of English: a Study in Subatomic Semantics*. MIT Press, 1990.
19. Chaïm Perelman and Lucie Olbrechts-Tyteca. *The New Rhetoric: A Treatise on Argumentation*. University of Notre Dame, 1958. Translated into English by John Wilkinson and Purcell Weaver in 1969.
20. Henry Prakken. An abstract framework for argumentation with structure arguments. *Argument and Computation*, 1(2):93–124, 2010.
21. Henry Prakken. On the nature of argument schemes. In Chris Reed and Christopher Tindale, editors, *Dialectics, Dialogue and Argumentation. An Examination of Douglas Walton's Theories of Reasoning and Argument*, pages 167–185. College Publications, London, 2010.
22. Iyad Rahwan and Leila Amgoud. An argumentation based approach for practical reasoning. In *AAMAS*, pages 347–354, 2006.
23. Iyad Rahwan, Bita Banihashemi, Chris Reed, Douglas Walton, and Sherief Abdallah. Representing and classifying arguments on the semantic web. *Knowledge Engineering Review*, 26(4):487–511, 2011.
24. Iyad Rahwan and Chris Reed. The argument interchange format. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 383–402. Springer US, 2009.
25. Chris Reed and Douglas Walton. Towards a formal and implemented model of argumentation schemes in agent communication. *Autonomous Agents and Multi-Agent Systems*, 11:173–188, 2005. 10.1007/s10458-005-1729-x.
26. Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
27. Glenn Rowe and Chris Reed. Argument diagramming: The Araucaria Project. In Alexandra Okada, Simon Buckingham Shum, and Tony Sherborne, editors, *Knowledge Cartography: Software Tools and Mapping Techniques*, pages 163–181. Springer, 2008.
28. Yuqing Tang and Simon Parsons. Argumentation-based dialogues for deliberation. In *AAMAS*, pages 552–559, 2005.
29. Bart Verheij. Dialectical argumentation with argumentation schemes: An approach to legal logic. *Artificial Intelligence and Law*, 11:167–195, 2003.
30. Douglas Walton. *Argumentation Schemes for Presumptive Reasoning*. Erlbaum, Mahwah, N.J., 1996.
31. Douglas Walton, Chris Reed, and Fabrizio Macagno. *Argumentation Schemes*. Cambridge University Press, 2008.
32. Adam Wyner, Katie Atkinson, and Trevor Bench-Capon. Towards a structured online consultation tool. In *Electronic Participation - Third International Conference (ePart 2011)*, volume 6847 of *Lecture Notes in Computer Science (LNCS)*, pages 286–297, Berlin, Germany, August 2011. Springer.
33. Adam Wyner, Trevor Bench-Capon, and Katie Atkinson. Towards formalising argumentation about legal cases. In Kevin D. Ashley and Tom M. van Engers, editors, *ICAIL*, pages 1–10. ACM, 2011.