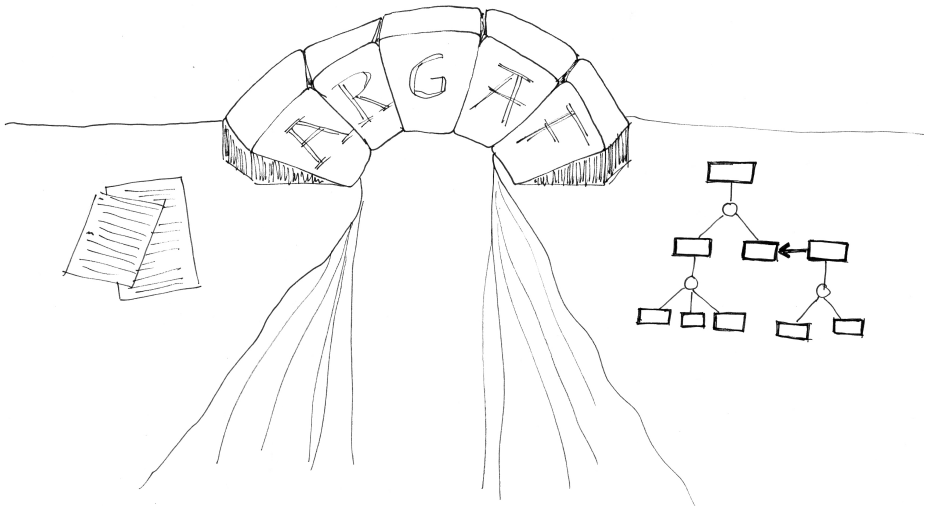
ArgAF:

The Argumentation Annotation Framework

A framework for annotating arguments
in filed legal cases
Establishing a bridge between documents
and the argument structures they contain



Bachelor thesis (15 ECTS)

Sara Veldhoen
3698661
Supervisor:
Prof. dr. mr. Henry Prakken

April 19, 2013

Acknowledgements

I wish to thank, first and foremost, my supervisor Henry Prakken for his supporting advice and useful comments. My thanks go also to Gerrit Bloothooft, who was involved as a supervisor in the first stage of this project. He has been very supportive in figuring out a direction for my research. I consider it an honor to have him as a second assessor as well.

I would also like to thank my fellow students Ruben Dulek and René van Gasteren for their dedication in proofreading my thesis in various stages of completion. Their remarks have significantly contributed to the quality of this thesis.

Furthermore, it has been a pleasure to discuss my data model with Jan Veldhoen, my father, whose critical investigations provoked new insights that lead to considerable improvements.

Lastly, I would like to express my appreciation for the organization of the thesis workshop on Februari 18th, which consisted of Alias[1] and a couple of PhD students among whom Jeroen Goudsmit. It was a supporting day and provided useful practical advice. Also many fellow CKI students [2] have been helpful by sharing their thesis writing expertise, thanks for that.

[1] aliasweb.nl

[2] www.uscki.nl

Contents

List of Abbreviations

AIF	Argumentation Interchange Format 25
AIF+	An AIF extension for dialogue 26
AIFdb	The AIF-based database design for the successor of AraucariaDB 31
AraucariaDB	The argumentation corpus (database) of the Araucaria project 31
ArgAF	Argumentation Annotation Framework 5
ArgAFdb	The database design of the ArgAF 31
argument_1	argument as a product 9
argument_2	argument as a process 9
CA	Conflict application 25
DCR	Data Category Registry 27
ECHR	European Court of Human Rights 5
GrAF	Graph Annotation Format 9
I-Node	Information node 25
LAF	Linguistic Annotation Framework 8
PA	Preference application 25
POS	Part-of-speech 38
RA	Inference application 25
S-Node	Scheme application node 25

Chapter 1

Introduction

The amount of data that is stored worldwide is growing, and it is growing fast. With it, the need for tools to process large amounts of data grows. This thesis focuses on textual data, in particular on documents containing argumentation in natural language. To process these documents, a specialization of natural language processing aimed at the exposure of argument structures is in development.

This thesis aims to contribute to argumentation research, by proposing a framework for annotating natural arguments as they occur in filed legal cases. Manually formalizing and storing the arguments as they are encountered in texts, is a first and necessary step in studying the automatic processing of argumentative texts.

1.1 Argumentation theory and Artificial Intelligence

Informally, argumentation can be defined as putting forward a claim and providing support for it. Classical argumentation theory, since Aristotle to present, studies the way argumentation is carried out by people and how to tell apart 'legitimate' from 'flawed' argumentation [1]. The difference between argumentation and formal reasoning, which occurs for instance in mathematical proofs or classical deduction, lies in the presence of defeasible rules. This means that a legitimate argument can be challenged by adding new information.

Argumentation has become a topic of increasing interest in the field of artificial intelligence. Different approaches have been taken to study and formalize everyday or defeasible reasoning, such as (and sometimes a combination of):

Evaluation of arguments Researchers aim to define a formal system that, given some argumentation, assesses the outcome of the dispute: does the conclusion hold, or its negation? The different entities making up arguments have to be formally defined, such as propositions, (defeasible) inference, attack relations between (parts of) arguments, and allocation of the burden of proof. In general, there is some conflict relation between arguments (pro and con some statement). A set of rules is then defined to resolve this conflict.

Reasoning agents Researchers create multi-agent systems where the agents can compose arguments based on their (individual or shared) knowledge base. The agents use a formal language wherein statements and inferences can be explicitly expressed. Some calculus must be defined to resolve the conflict between agents, so this area is somewhat related to the topic of evaluation of arguments. However, the invention of new arguments by the agents is a new focus in this area.

Visualizing arguments Another branch of study deals with argument diagrams. Given some argument, how is it best visualized? Software tools like Araucaria provide means to reconstruct arguments that are found in natural language text, based on different theoretical approaches like Walton's argumentation schemes. The arguments can then be visualized in graphs. Relations between (parts of) arguments can also fall within the diagram. This branch of study is interesting because it works on modeling not only artificial arguments, which satisfy known normative rules, but also natural argumentation.

Identification of arguments in discourse While the former three topics focus at formally defining argumentation concepts and relations, this more classical approach deals with arguments in written or spoken text. It aims to identify and analyze the argumentation as it occurs in natural discourse.

In itself, argument identification is rather a philosophical or linguistic (pragmatic) issue than an AI topic. However, the current information society demands more and more automatic processing of text documents. In order to apply the argument evaluation techniques that have been developed in other studies to natural argumentation, the arguments have to be extracted from the text and presented in a proper format. Since the identification and analysis of arguments in text is quite laborious, it is desirable to automatically perform this task. This research aims to contribute to this goal. The identification task is called *argument detection*, the analysis is initially reduced to a task called *argument classification*.

1.2 State of the art

To be able to train and test a system for detecting and classifying argumentation in text, an annotated corpus is needed: a collection of documents where the argumentation has already been classified. Some effort has already been put into this, which has resulted in the following two argumentation corpora.

The Argumentation Research Group at the University of Dundee has composed a corpus that is called AraucariaDB [2]. The corpus contains text material from diverse sources. Each argument is represented as a document in the corpus, with an annotation of the conclusion and premises. Furthermore, each document was assigned a scheme classification from a set of 'argumentation schemes' [3], a format for representing arguments that is described in section 2.2. The major drawback of this corpus is the absence of context: it consists of snippets of text containing an argument, rather than entire documents. The corpus can thus be used to train systems to *classify* arguments (for instance [4]), but not to *detect* them. After all, each fragment is a positive occurrence of argumentation, so there is no need to detect the borders of an argument within its context. Since the detection of arguments is considered an important task, the Araucaria Corpus has to be discarded for this purpose.

Mochales [5] worked on an argumentation corpus by having legal experts annotate filed legal cases from the ECHR (European Court for Human Rights). The framework she used for these annotations however, has some limitations. In chapter 4 this framework is described and evaluated. In [5], some first attempts towards the automatic detection and classification of arguments in text have been made using the ECHR corpus. Unfortunately, this corpus is until now not available for other research. It can therefore be assumed with reasonable certainty that it has not been used in other studies than those reported in [5].

No other mature argumentation corpora exist at the time of writing. The creation of a corpus is a first and necessary step towards the goal of automatic detection and classification of arguments. Given the scale of this research, it is not feasible to create such a corpus. By contrast, the requirements it should meet are investigated and an annotating framework to match these requirements is designed. This document presents a first version of this framework, which is called the ArgAF: Argumentation Annotation Framework.

1.3 Research goals

This research aims for the design of a framework for annotation of arguments in text documents. Each annotation framework should define what classifications (*annotation types*) exist and what features belong to each possible classification. Furthermore, it should establish some procedure for the annotators to follow.

There are two particular requirements for the ArgAF. Firstly, the annotations should be stored in an adequate format for machine learning. Secondly, the final classifications should be workable for further processing by some system that evaluates or organizes the arguments.

Many types of documents exist that contain argumentation. The ArgAF is designed to annotate filed legal cases, as did Mochales in [5]. The possibility of a generalization towards other document types is left open for further research.

Research Questions To achieve these goals, the following questions are asked:

(I.) What requirements should be imposed on the framework?

(II.) What are some of the problems with the existing argumentation annotating framework that was proposed in [5]?

(III.) How can existing argumentation models be related to natural argumentation?
Note that argumentation theories might be normative rather than descriptive.

- (a) What is a proper model of argumentation for this purpose?
- (b) What entities, such as propositions and attack relations, are defined in this model?
- (c) How are these entities represented in written text?
  Note that more than one sentence in a document may express the same proposition and vice versa, so the relation between assertions and propositions is not one-on-one. Furthermore, some argument entities might be implicit in the text.

(IV.) How should annotations be stored?

- (a) What annotation types should be defined?
- (b) What features should be assigned to these types?
- (c) What data format is suited for the storage?

Methodology This thesis contributes to the development of a corpus by examining the properties it should possess to be useful for empirical studies. It includes a large proportion of literature research on the one hand, and resulting therefrom a design phase.

The existing theories and current developments are investigated in order to reveal what limitations could be attempted to remedy. Because the problem concerns different area's (annotation and argumentation) the relation between the different existing approaches is studied in particular. Subsequently, an attempt is made to connect the concepts into a coherent whole. A data model and corresponding annotating framework are designed, to serve as a basis for actual corpus development.

1.4 The structure of this document

Chapter 2 introduces the foundations the ArgAF is built upon. The fundamentals of both argumentation theory and annotation are presented and some relevant concepts of these fields are introduced. Some observations about documents containing argumentation in general, and filed legal cases in particular, are made. Next, in chapter 3, research question I is answered: what requirements should be posed on the framework? A description is given of what the annotating framework should be capable of and how the quality of the framework should be assessed. In chapter 4 then, the annotating framework that was used by Mochales in [5] is evaluated to answer research question II. After a description of this framework follows a discussion of its limitations

and why it does not fulfill all the requirements that are posed on the ArgAF. In chapter 5 the data model is constructed to contain both the relevant argumentation concepts and the structural information to link this to the discourse that is recorded in the documents. This presents an answer to research question III. The data model is concreted in a relational database, and some remarks are made about the procedural aspects of the annotation process in chapter 6. This applies to research question IV: how should annotations be stored? In chapter 7 lastly, an overview of the answers to the research questions is presented. The implications of this research for the field of AI are discussed and some directions for future research are suggested.

Chapter 2

Foundations

In this chapter the fundamentals of annotation and argumentation are presented. Furthermore, some issues about argumentation as it occurs in documents are introduced.

2.1 Annotation fundamentals

An abstract Linguistic Annotation Framework (LAF) has been developed by an ISO committee in order to provide an international standard. This was considered desirable, because in linguistic research an increasing demand for commonality and interoperability was observed. Its outline was described in [6] and in [7] a description of both the LAF and the usage of it to represent the American National Corpus and its linguistic annotations are given. Although the nature of ArgAF is not purely linguistic, the LAF provides useful concepts that are described in this section.

Annotation The word *annotation* denotes two concepts: the classification of a piece of data (an annotation) or the activity of an annotator (annotating a corpus) [6]. An annotating framework correspondingly involves two separate parts, being the annotation scheme and the annotating process. The first is a formal specification of the classifications and features that can be assigned to the data, the latter is a procedural description of the task the annotators have to perform. Informally, an annotation is a 'classification' of a 'piece of text'. Other language data can be subject to annotation as well, but the focus here is on the annotation of written text. Thus, the following two tasks are involved [6]:

- **Segmentation**: the annotator needs to denote a piece of text or *segment*. The most obvious segment type is a *continuous segment*, which appears contiguously in the primary data. However, other possibilities are discontinuous segments (which are linked continuous segments), landmarks (points in the primary data) and even groups of *sub-segments* that constitute a *super-segment*.

- **Classification**: to each segment, the annotator allocates a type or *class*, e.g. 'Word' or 'Sentence'. It is common to also add *features* or properties to the classes, e.g. the attributes 'POS-tag' and 'Lemma' could be added to the class 'Word'. For each feature, the domain and allowance for 'null' values are given.

Pipeline architecture Linguistic annotation often involves several steps, where low-level annotations are used as input for a higher-level annotating step [7]. For example, in order to determine the parse tree of a sentence, we have to tag the words that occur in it first. This cascade-like process is called the *pipeline architecture*. In principle, the order of the stages is fixed, that is, the annotation of the low-level structure has to be completed before the higher-level procedure can start.

Stand-off annotation Annotations can be integrated in the document text as *in-line* tags, such as XML tags. This provides a direct relation between the text and the annotations. However,

the expressiveness of such tree-based structures is challenged by two tasks [9]. Firstly, multiple or alternative annotations require distinct trees that might be partially overlapping. Secondly, discontinuous units can only be represented by a tree if there is but one representation level. This is problematic, as multiple levels are involved in the representation in all non-trivial annotation tasks [9].

There are several solutions to this problem, including an approach using *stand-off* annotations. This approach is preferred by [9] and is also embodied in the LAF [6]. The annotation information is separated from the document text; it is specified how the annotations refer to text segments. The location of a character in the primary document can be referred to by giving its byte offset, as the byte representations of all characters have an equal length [7]. This works only for documents in certain encodings (such as Unicode), so an extra requirement has to be posed on the documents. An atomic annotated text segment is thus represented by its start offset and its end offset. Alternatively, an annotation can be recursively defined as referencing two existing annotations [7]. The notion of stand-off annotations allows for a clear distinction between the content and the format [7].

Representation Annotations are rendered in a certain format, which is called their *representation*. The representation is independent of the content, the same annotation may be represented in different formats and the same format may accordingly represent different annotations [6]. According to the LAF design, a mapping must exist from the representation that is used and the underlying *abstract data model* [7]. A rigid *dump format*, which should be isomorphic to the data model, accomplishes this mapping. The primary intention for this dump format is to be machine readable.

Data Model The model is a schematic, formal description of the annotating information. The main principle here is distinction between the *structure* and the *content* information [7].

The annotations are suggested to be modeled as a directed graph. This model is universally used in general-purpose annotation formats [7]. A more elaborate description of modeling annotations as a graph is presented in [8], which introduces the GrAF (Graph Annotation Format). In the GrAF, text segments are modeled as edges between pointers to the primary data. Annotations are modeled as nodes. The content information (type and possibly features) is stored in the annotation node. The structural information is represented by edges from the annotation nodes to text segments or to other annotation nodes (in the case of annotations over other annotations).

2.2 Argumentation fundamentals

The word *argument* too has different connotations. It can be interpreted as a process where persons (or agents) make statements and give reasons that either support or attack statements; it is referred to as argument$_2$ in this case. The argumentative structure that is the result of such a process is referred to as argument$_1$. These names are not very descriptive, but are widely used in literature [14]. In an argument$_1$, an inference is drawn from premises to a conclusion, with the characteristic that the inference may rely on either defeasible rules or classical deduction. Just like other inference structures (such as classical deduction) arguments resemble tree structures, and indeed are commonly modeled as trees.

The annotation task the ArgAF is created for consists of extracting the complete, formal argument$_1$ structure from a description of discourse: argument$_2$. This relies on interpretation of the intentions and beliefs of the participants. However, each interpretation slightly reduces the reliability of the result. Therefore, only interpretations should be allowed that either improve the argument$_1$'s consistency or really accommodate the annotator's work. This is a leading consideration in the following discussion.

Different argumentation theories have been established by researchers with various perspectives. This has lead to theories treating some part of argumentation in detail, but leaving other concepts unspecified. To determine the theoretical basis for the ArgAF, aspects from different argumentation theories will be combined.

2.2.1 Inference Rules

The inference rules in argumentation can be deductive or defeasible. Deductive inferences are monotonic, which is to say: given the truth of the premises, the conclusion must hold. Defeasible rules are different in nature: given a set of premises, *presumably* the conclusion is true as well. Adding extra premises may however show the conclusion to be false [10]. The notion of non-monotonic consequence has been the object of study called 'non-monotonic reasoning', which is a subfield of Artificial Intelligence.

Argumentation Schemes Different approaches exist regarding non-monotonic reasoning, one major theory proposes *argumentation schemes* to model argumentation. A description of this theory and its implications can be found in [3]. There is no agreed-on set of argumentation schemes, but an extensive list of schemes is given in [3, chapter 9]. An example of an argumentation scheme is given in figure 2.1.

ARGUMENT FROM POSITION TO KNOW

Major Premise: Source a is in position to know about things in a certain subject domain S containing proposition A.
Minor Premise: a asserts that A is true (false).
Conclusion: A is true (false)

Critical Questions

CQ1: Is a in a position to know whether A is true (false)?
CQ2: Is a an honest (trustworthy, reliable) source?
CQ3: Did a assert that A is true (false)?

Figure 2.1: Argument from position to know, as described in [3, page 309]

An argumentation scheme describes what propositions can make up an argument. Furthermore, each scheme comes with a set of *critical questions* that indicate how an argument$_2$ that is an instance of this scheme can be questioned, in order to reveal its weaknesses.

The premises embody the essential elements of the argument, whereas the critical questions serve as a reminder of the assumptions that are made, thus revealing ways to probe the weaknesses of the argument [3]. Some of the critical questions merely question the premises of the scheme, such as *CQ1* and *CQ3* in figure 2.1.

There are also critical questions that point to *assumptions* (note that in literature also the word *presumption* is used for this phenomenon). Assumptions resemble premises, but are not required in the composition of a complete argument: they are mostly implicitly assumed to be true. Another way to look at assumptions is that they point at emphexceptions: situations in which the inference should not be drawn, even if the premises are true. *Q2* in figure 2.1 is an example of such a question.

The ArgAF will rely on argumentation schemes to model the arguments. It has been argued that argumentation schemes provide a proper basis for modeling natural argumentation [11]. According to [5], an argumentation scheme can be assigned to any argument$_1$ occurring in natural

argumentation. Furthermore, the argumentation schemes are simple patterns that allow for automatic processing and moreover are an intuitive formalism that is easy to work with for human annotators [5]. These characteristics are beneficial for the ArgAF as well.

One objective of the ArgAF is to be compatible with argument processing tools. As argumentation schemes are a common way to model argumentation, this formalism is presumed to be compatible with many tools. Furthermore, all arguments that are stored as instantiation of an argumentation scheme can be translated in a more general argument form, by simply discarding the specific roles and only distinguishing the premises and conclusions. Therefore, this formalism can also be used for tools that are less sophisticated, whereas a translation in the opposite direction would be impossible. Of course the stored annotations would not be compatible with tools taking a fundamentally different approach.

2.2.2 Discourse

Studying the relation between argument_1 and argument_2 exposes some interesting issues, which are discussed below.

Assertions Although an argument_1 can be considered to be built up from *propositions*, it is important to realize that these propositions are *assertions* in argument_2. An assertion is a speech act involving a proposition, but also the speaker and the circumstances such as time and place, which constitute the (dialectical) context of the assertion.

The relation between statements in the text and their meaning in the argument_1 structure is not one-on-one: multiple sentences may refer to the same proposition. Either verbatim (the same phrase occurring in different places in the text) or semantically: different phrases denoting the same proposition. It may even occur that the same phrase denotes different propositions in different places, due to the scope of anaphora for example.

Furthermore, assertions can be either atomic or complex. Sometimes a complex statement needs to be deconstructed, as its parts play different roles in the argument. The deconstruction of statements is not a trivial task, as the semantics of natural language conjunctions may vary in different situations and when discarding conjunctions, the richness of their semantics might get lost. Therefore, deconstruction should only take place if it is mandatory for a good reconstruction of the argument, in other situations the assertions should be kept intact.

Enthymemes Sometimes in argumentative discourse, premises (and even the conclusion) are not asserted but left implicit. The argument_2 is then commonly referred to as an *enthymeme*. It has been argued that most everyday arguments_2 are enthymematic [5, page 12]. However, the argument_1 structure is supposed give a complete view of the argumentation. Discovering the complete argument_1 structure of an enthymematic argument_2 is not a trivial task: filling in the implicit premises and/ or conclusion rely on *interpretation* of what the arguer meant, which might be wrong [3].

Time Argument_2 takes place in stages, the agents take turns reasoning towards or against some statement. The proceeding of discourse is monotonic, which is to say: an assertion cannot be undone. The sequence of speech acts can be expressed by transitions from each stage to the next.

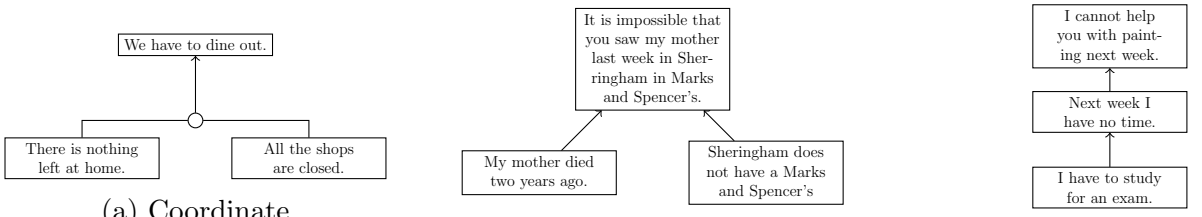
The documents the ArgAF is intended to annotate typically contain a resumé of such discourse, presenting the product of the argumentative process (all the reasoning structures, possibly with support and conflict relations between them) rather than displaying the process in a chronological order. Quite often there is no explicit information about the progress of the discourse in these document, nor do we need this information to be stored.

However, to accommodate for other document types an extension may be formulated, incorporating the stages of the discourse.

2.2.3 Relations between arguments

Natural arguments rarely appear in isolation. Relations between arguments$_1$ are either supportive or conflicting.

Support relations Natural argumentation is often a complex structure built up from several arguments$_1$ working together to substantiate a conclusion. A distinction is made between three kinds of support relations. Firstly, arguments with one inference, which will be called *simple arguments*, can have several premises that are *coordinate*. This means the premises are interdependent to yield a conclusion. Secondly, multiple independent arguments that support the same conclusion occur in *multiple* argumentation. Although multiple arguments may share premises, the arguments can be completely unrelated except for their conclusion. Lastly, an argument supporting a premise of another argument is an occurrence of *subordinate* argumentation. In figure 2.2 each of these structures is presented in a diagram. These are examples from [12] that are presented in [5, page 17] to clarify the three structures.



(a) Coordinate argumentation

(b) Multiple argumentation

(c) Subordinate argumentation

Figure 2.2: Support relations

Conflict relations Conflict relations between arguments$_1$ occur when an agent attacks an argument$_2$ that was put forward. There are divergent theories about how conflicts between arguments should be formalized and how their relative strength is computed. The ArgAF should give a very general formalization of the conflict relations, leaving the possibility of different interpretations to further processing.

An influential theory is the so-called 'three way hypothesis' (a term that is employed in [3]). To be compatible to argument processing tools, the concepts in this theory need to be modeled in the ArgAF. According to the three way hypothesis, three types of attacks can be distinguished [13]:

1. attacking a premise: *undermining*,
2. attacking a conclusion: *rebuttal*, and
3. attacking an inference: *undercutting*

While the first two attack types rely on a conflict relation between assertions, the third type requires a more sophisticated approach.

One way to model undercutting defeat in the argumentation schemes formalism uses the critical questions that are specified for each scheme. Remember that some of the critical questions point to exceptions in which the inference should default. These exceptions can be formulated as propositions. An argument with such a proposition as its conclusion can be used as an undercutting attack.

The assumption that the critical questions are complete, that is, specify all possible exceptions, is at least controversial. The ArgAF will adopt this method for modeling undercutting defeat nonetheless. Application of the ArgAF to real argumentation should reveal the fitness of this theory in practice.

Questioning An agent can also question the premises of an argument$_2$, rather than oppose them. This is a special kind of conflict relation, and may have a different effect on the status of an argument$_2$ than an undermining attack: there is no implication that the questioner is against the questioned proposition.

Questions are only meaningful in argument$_2$, and are not part of argument$_1$, the product of the discourse [14]. This can be illustrated with a dialogue (argument$_2$) where agent 1 makes some statement P_1. Another agent may ask "why P_1?", after which the first agent provides a reason P_2 from which P_1 may be inferenced. The corresponding argument$_1$ will be "P_2, therefore P_1". The question can no longer be retrieved from this information.

Relations between assertions Relations between assertions can serve to infer relations between arguments. Recall, for instance, that an undermining attack has a conclusion that opposes a premise of the attacked argument. Rather than the conflict between the arguments, the conflict relation between the statements is stored.

Classical contradiction (p and $\neg p$) is however not sufficient: other types of conflict between statements exist. An extensive description of theories of contraries and opposition is given in [3, section 7.2]. For example, the statement S1:"The grass is green" is opposed by S2:"The grass is red". A possible solution is to add an inference to S3:"The grass is not green" from S2, resulting in a classical contradiction of S1 and S3. However, this adds a new statement based on interpretation, whereas storing the fact that the statements S1 and S2 are (somehow) conflicted remains closer to the actual discourse. Since conflict relations need to be stored already, that could include other types of conflict as well.

Modeling different types of conflict relations based on the meaning of the assertions, stretches beyond the purposes of this first version of the ArgAF. It suffices to store the relation as it is classified by the annotators, but it is surely interesting to study these relations for a future extension.

Also support relations between arguments can be expressed in terms of agreement between their premises and/ or conclusions. And again, there are other types of agreement than statements denoting exactly the same proposition. For instance, the statement S4:"The grass is green and the sky is blue" supports S1. At first view, the same two approaches can be taken: either to explicitly store the agreement relation between S4 and S1, or to add an inference from S4 to S1. In this case however, the approaches coincide: no new statement is added in either case, only a connection ('agreement relation' or 'inference step') is established between the existing statements. At most, new inference rules have to be defined to account for the different possible types of agreement between statements.

2.3 Documents containing natural argumentation

Natural argumentation can occur in very different circumstances. The number of agents participating, the modality of the discourse (whether it is in writing or spoken), the formality of the setting and the goal of the discussion are features that determine how the argumentative process is carried out.

2.3.1 Mismatch between document and discourse

Analysis of natural argumentation is often performed to a *record* of argumentative discourse, not the actual discourse itself. It may concern audio or video material but mostly written text. In the latter case, an oral or written debate is saved in a *transcription*.

Any transcription of discourse reduces information, such as the time and place of an utterance, the speaker or his gestures. Even if the discourse was itself in writing, information about

the circumstances (such as authorship) might be lost. The loss of information is not always a problem, as not every aspect of the situation is meaningful to the argument$_2$ itself. However, this information may be rather helpful in the reconstruction of the argument$_2$. And some other information is essential to the argumentation.

In many cases, a transcription also adds information. In a literal transcript of oral discussion, the transcriber adds punctuation, and may restructure a sentence to make it grammatically correct or easier to understand. When the author of the transcription is summarizing the discourse, he might change the order of the clauses to clarify the structure of an argument$_1$. The transcriber can even (intentionally or not) express his own sentiment by using certain expressions. These adjustments can be helpful in the reconstruction of the argument$_1$, but might also be misleading as the interpretation of the transcriber might not be completely adequate and because information about the argument$_2$ might be lost.

The adding and discarding of information alters the information contained in the document, to good or to bad. In spite of this, the document is all there is, its content is the starting point for the annotation. Any doubts about it exist at another level and should not be included in the annotation.

2.3.2 Argumentative discourse and its argument structure

The documents the ArgAF is concerned with contain a resumé of argumentative discourse: argument$_2$. It contains both descriptions of speech acts and an interpretation of the way these are part of the argument structure: argument$_1$. Consider for instance the following sentence:

> *"On the basis of the above Complainant maintains that the Disputed Domain Name was both registered and used in bad faith."*
>
> (WIPO case d2002-0801, [1])

This sentence expresses a locution (speech act) with a proposition P_1 ="the Disputed Domain Name was both registered and used in bad faith" that is uttered by a party E_1 =the complainant. Furthermore, it relates this utterance to statements that have been made earlier in the text, which illustrates its embedding in the argument structure. The discourse information can be annotated directly, whereas the argument information is more indirect, it merely offers an indication of how the reconstruction of the argument$_1$ should take place. Note that this is an interpretation of the intention of the speaker. Also the propositions in the argument network are interpretations of beliefs of the speaker, rather than his literal statements, although these may coincide.

The document contains both information about the argument$_1$ and argument$_2$, but this information is rarely complete with respect to either domain. Reconstructing information is risky, but it makes sense to make an interpretation to create a complete argument$_1$ network, whereas reconstructing the discourse is less relevant.

2.3.3 Discriminating discourse information and argument structure

Imagine the following conversation:

Artistotle:	A1: Socrates is mortal.
Plato:	A2: Artistotle says that Socrates is mortal.
	A3: Aristotle is always right,
	A4: therefore Socrates is mortal.

It seems perfectly reasonable to connect the first sentence Plato utters (A2) to the first clause (the assertion of A1), as Plato quotes the words by Aristotle. But now consider the following, slightly altered conversation:

[1] http://www.wipo.int/amc/en/

Artistotle:	B1: Socrates is mortal.
Plato:	B2: Artistotle says that Socrates is a turtle.
	B3: Aristotle is always right,
	B4: therefore Socrates is a turtle.

Again, Plato quotes Aristotle in B2, but in this case Plato is wrong: he *believes* Aristotle said that Socrates is a turtle, but he misheard the statement (B1) Aristotle made. The statement B2 is completely new in this case, it cannot be connected to an earlier statement. This is a somewhat artificial example of course, but situations with people believing someone said something, while in fact he said something (slightly) different are easy to imagine.

The above example shows that the information *about* the discourse is in another domain than the information contained *within* the assertions. The truth of the statements in the first domain is assumed by the annotator: the document text is the starting point. The transcriber might have been wrong as well, but that is not of concern for the annotation task. The statements in the the latter domain, however, are part of the argument structure that is the subject of annotation. In fact, their truth is of no importance for the annotation task.

Because of this, entities in the argumentation should never refer directly to concepts in the communication level. The documents to be annotated contain information about both domains. What the parties said will be called 'locutions' (speech acts), these can be annotated. What the parties meant (from an argumentative viewpoint) is an interpretation of the locutions. This is where propositions, inferences, conflict and support relations are situated.

Chapter 3

Requirements

From the findings of chapter 2, it becomes clear what requirements should be met by the ArgAF. There are also demands that may only prove to be satisfied after application of the framework.

3.1 A priori requirements

Content information Usually, each annotation has a type and a (possibly empty) set of features. However, there is much more to the content information in the ArgAF: the annotations may relate to nodes in an argument graph, which is itself a rather complex structure. Determining this relation is what the annotation task is really about.

Many documents simply do not offer enough information to recover the complete argument_2 (the discourse structure), but neither is the argument_1 structure explicit. The ultimate task for the annotator is to reveal the complete argument_1 structure by interpreting the partial information about both structures. As has been argued, the possibility for storing annotations that have no meaning for the argument_1 structure should be retained.

Remember that the LAF demands the definition of a data model that specifies the structures to be annotated. Surely, the following concepts that make up argument_1 structures need to be represented:

- Statements (propositions) in the role of premises, conclusions and exceptions.
- Inferences drawn according to argumentation schemes. Different sets of argument schemes could be defined, perhaps dedicated for a specific domain. An argumentation scheme must specify which conclusions may be drawn from what premises. In addition, an argument scheme may specify what exceptions cause the inference to default.

  Basic deductive inferences should not be forgotten: not all argumentation follows defeasible rules. Remember, furthermore, that agreement between statements is stored in inferences, which may require other schemes to be added.
- Conflict relations between statements. As has been argued in section 2.2, different types of conflict relations can occur and need to be stored. The conflict relations between arguments can be derived from these. The specification of exceptions as statements allows undercutting attack to be derived as well.

There are also concepts that exist primarily or exclusively in the argument_2 structure. These need not be modeled for the core task, but might prove helpful for the annotation task as well as for the future machine learning.

- Locutions: primarily, storing assertions is useful as these are direct suppliers of statements for the argument_1 structure. Other locutions include questions, which do not fulfill a role in the argument_1 structure but may have an important function in argument_2, for instance in the role of critical questions.
- Enthymemes: an argument_1 structure is assumed to be 'complete' regarding the argument scheme. Thus, the notion of enthymemes resides in the argument_2 scope. Because of that,

this property can be derived: if there is no connection from a statement in an argument$_1$ structure to the textual data, the corresponding argument$_2$ is enthymematic.

Structural information A demand that is inspired by the LAF is to separate the structural information (segmentation) about the annotations from their content information (classification). It might seem that this requirement stretches beyond the purposes of the ArgAF, since for the core task (the restoring of the argument$_1$ structure) the annotation structure needs not be very complex. Only 'statement'-annotations have to be stored, as these are the only annotations that link to the argument network.

However, other annotations (such as locutions) may help the annotator with this task, and indeed might also be helpful in the task of automatically determining the argumentative role of text fragments. Indeed, if an external system were able to parse the documents and annotate their discourse structure, this might serve as a basis for some future argument reconstruction tool. Far more complex annotation structures with various levels could thus be useful, so it is desirale to indeed make the distinction between the structural and content information.

Storage The ArgAF is intended to create argumentation corpora that are usable for both *machine learning* of argument detection and classification, and *argumentation processing* tools. This creates a specific constraint on the representation: the annotations should be accessible at both ends.

Furthermore, the annotations must be stand-off, which is to say they are to be stored in a file separate from the documents themselves. This retains the possibilities of comparing or combining different versions of annotations for the same document.

Lastly, the annotations should be stored in a format that is isomorphic to the data model underlying the framework: the so-called 'rigid dump format'.

Procedural aspects There needs to be a specification of how the annotating should be carried out. In order to obtain the argument$_1$ structure, several subtasks for the annotators can be determined, such as the annotation of discourse structure. Instructions for these tasks need to be provided. Furthermore, the procedure should dictate whether each of these tasks is mandatory and whether they should be performed in a fixed order (pipeline architecture) or not.

3.2 Evaluation

It is beyond the scope of this research to have annotators actually use the ArgAF, let alone to use the annotations for machine learning or argument processing. In fact, these three tasks should determine the quality of the ArgAF, as the quality of any annotating framework is assessed by evaluating its performance in practical application.

Annotation quality The stability and reproducibility are quality measures for all annotating frameworks [15]. The former refers to the extent to which one single annotator produces the same annotations at different moments and is a prerequisite for reproducibility. The latter is the extent to which different annotators produce the same result, which reflects the conceptual consistency. These measures can only be computed after the ArgAF has been used to annotate documents, by using the Kappa coefficient for example: the agreement between annotators is then compared to a random distribution.

Only after several annotators have used the framework to annotate the same documents, can the stability and reproducibility be assessed. The Kappa measure can then be computed as well, but there is no gold standard to compare it to. We might compare the value to the Kappa measure

that is assessed for the framework that was used by Mochales in [5], if the framework is used for the same set of documents.

Performance Besides the quality of the annotations, the usability of the database for the two tasks it is designed for is of course the most important performance measure. This too can only be assessed after a reasonable number of documents have been annotated, and furthermore these annotations have been used for either task: machine learning of argument detection and classification, and processing arguments by different tools.

Chapter 4

Evaluation of Mochales' Framework

Mochales' research [5] was aimed primarily at detection and classification of argumentation in legal cases. However, as she learned there were no serviceable corpora available, the creation of a corpus became a major part of her project. In [5, chapter 5], Mochales presents the framework that was used to annotate her ECHR (European Court of Human Rights [1]) corpus. A description of this framework is presented here, after which some unclarities and concerns are pointed out. To conclude, the framework is evaluated in the light of the requirements of the ArgAF.

4.1 Description of the framework

Mochales describes the framework in two ways. Firstly, she states the task with sentences as a starting point.

> *"[...] for each sentence it should be necessary to determine if it is part of the argumentative process or not. If the sentence is part of the argumentative process then, it has to be decided if it is part of the final decision or the set of arguments to achieve the final decision. If it is part of an argument its function in the argument has to be determined. Moreover, if two or more arguments are related it should be indicated."*
>
> ([5, page 60])

After this global description, an explanation of the steps to take is given. First, the arguments in the text should be detected. Then each argument has to be classified (assigned a type). After this, the elements (premises and conclusion) of the argument have to be classified. The result of these steps is:

> *Each detected argument is represented by the following information:*
>
> - *Group of premises:* $P_i, i = [0..n]$
> - *Conclusion:* C
> - *Type of argument:* $T_a = [1..m]$
> - *Reported/ Non-reported*
>
> ([5, pages 64-65])

This results in several isolated coordinate arguments. The relations between the arguments can be derived from this information. Indeed, arguments that have the same sentence as conclusion are multiple and if a sentence that is the conclusion of some argument serves as the premise of another argument, the arguments are subordinate [5].

The data model underlying this framework is believed to correspond to the entity relationship diagram that is displayed in figure 4.1. Rectangles, diamonds and ellipses represent entities, relationships and attributes respectively.

[1] http://www.echr.coe.int

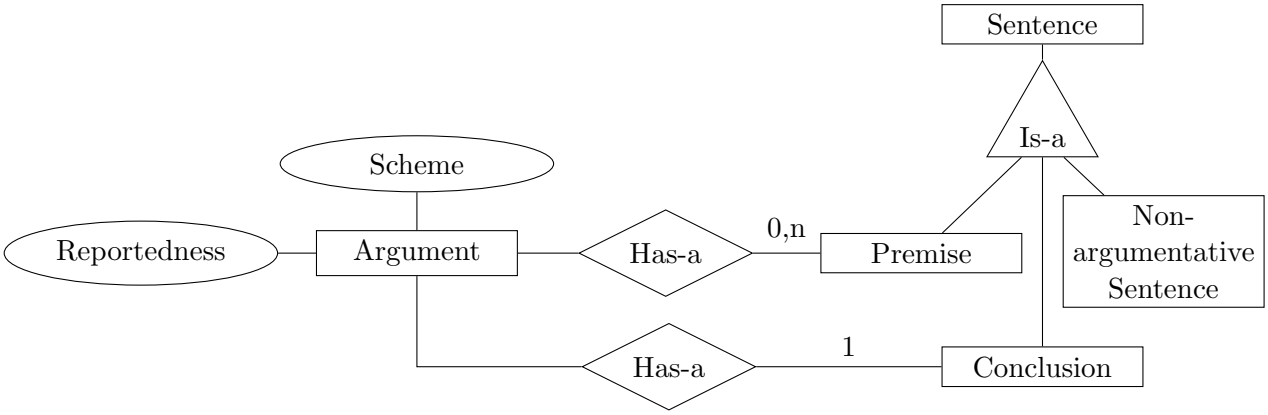


Figure 4.1: A diagram of the data model that was used by Mochales

The procedure of the annotation is described as:

- *Lecture for general knowledge*
- *Identification final decision*
- *Identification of single arguments*
  - *Identification conclusion*
  - *Identification premises*
  - *Identification type*
- *Recursive till all connected to final decision*
  - *Connection between single arguments*
  - *Integration in super-argument*

([5, page 66])

4.2 Considerations

This framework has proven itself useful in the studies Mochales has conducted [5]. However, it is based on assumptions that require some attention. This section discusses some questionable claims and indicates the decisions that were made and their effect on the results.

4.2.1 Granularity

Annotations can exist at several levels. For many machine learning algorithms, a decision has to be made in advance about the level of granularity. When working with a classifier, for instance, the training data is broken down into *documents*, which are the objects to be classified. For natural language applications, these levels can be paragraphs, sentences, words, etcetera.

For the annotation of arguments, Mochales chose sentences to be the level of granularity:

> *"[..] a suitable annotation should annotate all the sentences in each document and for each sentence it should be necessary to determine if it is part of the argumentative process or not."*
>
> ([5, page 60])

However, she does not provide reasons to believe that a sentence expresses exactly one conclusion or premise. The choice of sentences as granularity level for the annotation of argumentation is argued by means of the following two cases.

1. In [5, section 4.2.5], a reasoning pattern is described that occurred a lot in the corpus: a form of *argument from precedent* where the entire argument (premise and conclusion) is presented in a single sentence. The connection between the two parts is represented by a keyword. The following example is given:

   > *"It is furthermore established that the burden of proving the existence of available and sufficient domestic remedies lies upon the State invoking the rule (cf. Eur. Court H.R., De Jong, Baljet and Van den Brink judgment of 22 May 1984, Series A no. 77, p. 18, para. 36, 11.5.89, D.R. 61, pp. 250,262)."*
   >
   > ([5, page 56])

   In this example, a single sentence contains both the premise and the conclusion:

   > **Premise:** *"(cf. Eur. Court H.R., De Jong, Baljet and Van den Brink judgment of 22 May 1984, Series A no. 77, p. 18, para. 36, 11.5.89, D.R. 61, pp. 250,262)."*
   > **Conclusion:** *"It is furthermore established that the burden of proving the existence of available and sufficient domestic remedies lies upon the State invoking the rule."*
   >
   > ([5, page 56])

   Mochales solved this problem by modifying the corpus. She used an automatic process to separate sentences such as this one into two separate sentences, based on the keywords {*"cf."*, *"see"*,*"see eg."*, *"see also"*}. In this way, the annotators could properly classify the two parts of the sentence as premise and conclusion.

   Although this method might suffice for the ECHR corpus, other situations are conceivable where the conclusion and premise are contained within the same sentence.

2. Consider the following sentence, which was taken from ECHR case no. 50064/99, Girardi v. Austria:

   > *"The court therefore finds that the overall length of the proceedings cannot be regarded as reasonable."*
   >
   > (ECHR case no. 50064/99, Girardi v. Austria)

   According to Mochales, this entire sentence should be classified as being the conclusion of an argument. However, it expresses more than the actual conclusion, which is only a part of the sentence:

   > *"the overall length of the proceedings cannot be regarded as reasonable."*

   while the words

   > *"The court therefore finds that"*

   merely express the reasoning step, and furthermore provide information about the entity expressing this conclusion.

Therefore, the granularity of an argumentation annotation framework should be less coarse than sentences. Unfortunately, there is no obvious alternative. It may not always be possible to annotate a known linguistic unit as 'statement': statements may be sub-sentences that are linked with connectives, but they may also be interleaved.

4.2.2 Procedure

Mochales distinguishes three tasks (or aspects), being

- *Detection and classification of arguments*
- *Classification of elements of an argument*
- *Relations between arguments*

([5, page 60])

She implies, also in the description of the procedure of annotation, that these tasks should be performed in this order. However, detecting a complete argument at once seems quite a hard task. It might be easier to first detect the statements that can constitute arguments and then work out how they are related to an argument scheme and how they form a complete argument. It is even possible to imagine a process of switching back and forth between these steps (or levels) to reveal the proper structure of the argument. The suitability of any order of the steps has yet to be demonstrated and it is until now not clear what is the proper way to accomplish this task.

4.2.3 Argument theory

The choice of argument schemes as a starting point for the annotation is convenient as it is an intuitive system. There is nothing wrong with that, but keep in mind that this is not the only possibility and that this choice may very well affect the information contained in the annotated corpus.

Besides, Mochales points to another study by Walton [16], where the argument schemes are considered to be a hierarchical system with each scheme being a subcategory of another. By allowing annotators to choose the proper annotation of an argument from this hierarchy, the agreement between annotators per level (in the hierarchy) might improve. [5]. This idea deserves further investigation, it is added to the suggestions for further research in chapter 7.

4.2.4 Reportedness

In the ECHR cases, some of the arguments are 'reported', which indicates that the arguments have been previously filed to the court, while other arguments are 'non-reported', which are the arguments put forward by the court itself. This reportedness status is not an intrinsic property of the arguments themselves, but rather of the role they have in the document. It is important to be aware of this distinction, and to make a deliberate decision about the way this is modeled.

4.2.5 Critical Questions

Recall the critical questions that were introduced in section 2.2, belonging to the argumentation schemes. Mochales is not clear about the role of critical questions:

> *"Critical questions are important tools for the evaluation of argumentation when working with argument schemes. In the ECHR corpus they figure as premises, either implicit or explicit, in the argumentation of both parties, either in anticipation of critical questions from the counter-party (this is usually the case for the applicant's argumentation) or in answer to the critical questions from the counter-party (which is usually the case for the government)"*
>
> ([5, page 64])

Mochales points out that critical questions function as premises in the ECHR corpus, but she does not explain how this works and how annotators should deal with critical questions. Recall from section 2.2 that some critical questions point to assumptions or exceptions, from which conflict relations between arguments can be derived.

4.2.6 Relating arguments

Mochales states that relations between arguments can and should be extracted from the information gathered in the earlier stages of the procedure. But the relations between arguments are defined as relations between their parts:

> *"If the premises of two arguments maintain a coordinate of multiple relation, both will have the same conclusion"*
>
> ([5, page 65])

Because of this, it is extremely important to distinguish propositions from sentences, as two sentences may express the same proposition and thus a relation holds between the arguments wherein this proposition has a role, even though the sentences occur in different places in the document.

4.3 Evaluation

This section investigates the extent to which the framework that was described in this chapter fulfills the requirements that were posed for the ArgAF in chapter 3.

Content information The ArgAF requires the data model to represent statements, inferences, and relations between statements. No data model is explicitly defined to serves as a basis for Mochales' annotating framework. An attempt was made to deduce the data model from the descriptions, it is displayed in figure 4.1.

The diagram shows that not statements but sentences are considered the basic units, which has already been criticized. The inferences are represented as arguments with an argument scheme in the diagram. However, no information is stored about *how* the scheme is instantiated by the sentences. Relations between statements (or sentences) are not represented in the diagram. Relations between arguments are derived from the fact that they have the same sentence as premise or conclusion, but this does not suffice as different sentences can express the same statements.

Structural information To store the content and structural information, the document text is broken down into sentences with a unique index. The annotations reference such an index, which is their structural information. In particular, an argument annotation references the sentence indexes of its premises and conclusion. The same sentence can be referenced by different arguments.

The structural information is thus separated form the content information, as the ArgAF requires. However, there is no way to store intermediate annotations that could ease the annotation task.

Storage Mochales' framework partially fulfills the storage requirements, as she does store the annotations in a separate document. However, she did not explicitly specify a data model that defines how the arguments are represented internally. It is therefore uncertain if and how the argument structure could be extracted and used by other tools than the machine learning algorithms it was created for: classification of sentences, and parsing of sequences of sentences.

Procedural aspects Mochales is clear about the subtasks and provides a description of the way and order in which they should be performed. However, it is uncertain whether the decisions that were made, lead to the best possible annotation results.

Discussion It is important to differentiate between the model and the text. The structure of the data model should be explicitly defined. Annotations function as a bridge between the text and the data model. A framework should aim for enough informativeness in the annotations, without losing generality and while keeping an eye on the clarity of the definitions.

The model contains the argument, which is a set of propositions and their relation expressed by the reasoning step, represented by an argument scheme. The text contains sentences describing the utterances by the parties. Their utterances contain propositions. The propositions should be distinguished from the occurrence of words expressing them.

Chapter 5

Abstract data model

The purpose of this chapter is to present an abstract data model that represents the argumentation concepts described in 2.2 and the structural information about the annotations. The model was founded on the AIF ontology and the GrAF.

5.1 AIF: Argumentation Interchange Format

A group of argumentation experts has created an interchange format capable of expressing the various argumentation concepts that are described in the concerned fields. The purpose was to define a single markup language that would account for different kinds of theories and applications, so that ideas and results could be interchanged. In [17] a first draft version of the AIF is presented, which is a rather abstract data model. A later version is presented in [18], which is extended with ideas from [19] to incorporate the argumentation schemes theory by [3].

5.1.1 Extended AIF core ontology

Here follows a description of the AIF as it was presented in [18]. All concepts are grouped into three categories: arguments, communication, and context. The arguments ontology will be reffered to as the 'upper layer', the 'forms layer' belongs to the context category. These two layers are connected by 'fulfills'-relations between their concepts.
In the **upper layer**, arguments are modeled to be networks, with argument entities as nodes in a directed graph. There are two kinds of nodes: I-Nodes and S-Nodes. An I-Node contains *information*, which is a proposition in the argument. An S-Node represents the *application* of a scheme. There are three types of S-Nodes: inference application (RA), preference application (PA), and conflict application (CA).

Directed edges may connect I-Nodes to any S-Nodes or S-Nodes to other S-Nodes. If there is an edge from N_1 tot N_2, then N_1 is said to *support* N_2. The semantics of the edges can be inferred from the types of Nodes they connect, there is no need to explicitly mark the edges. For instance, if an RA-Node N_1 supports an I-Node N_2, the edge means that N_2 is the conclusion that is inferred in N_1.

The **forms layer**, contains the schemes that are *fulfilled* by the S-Nodes. There are three types of schemes accordingly: inference schemes, preference schemes, and conflict schemes. The extension by [19] aims to integrate the argumentation scheme theory by [3]. An argumentation scheme schematically describes the roles of statements that make up an instance of the argument: its premises and conclusion. So-called *descriptions* are added to the form layer, so that an I-Node can be said to *fulfill* such a role. Note that in [19] the descriptions are called 'F-Nodes', but that practice obscures the distinction between the two layers as nodes are the entities in the upper layer.

A 'uses'-relation may exist between two forms. Recall for instance the 'argument from position to know' (figure 2.1). Its scheme in the AIF *uses* two premise descriptions and one conclusion description.

As regards the critical questions, a distinction is made between two types. Questions that capture assumptions have a corresponding *presumption description* in the AIF, which is *used* by the scheme. Critical questions that capture exceptions on the other hand, are contained in a *conflict scheme* which is said to be *used* by the inference scheme as well. An undercutting attack (attacking the inference step) at an argument fulfilling some scheme S is thus represented by a CA-node, supported by an I-Node that fulfills some exception description belonging the the scheme S.

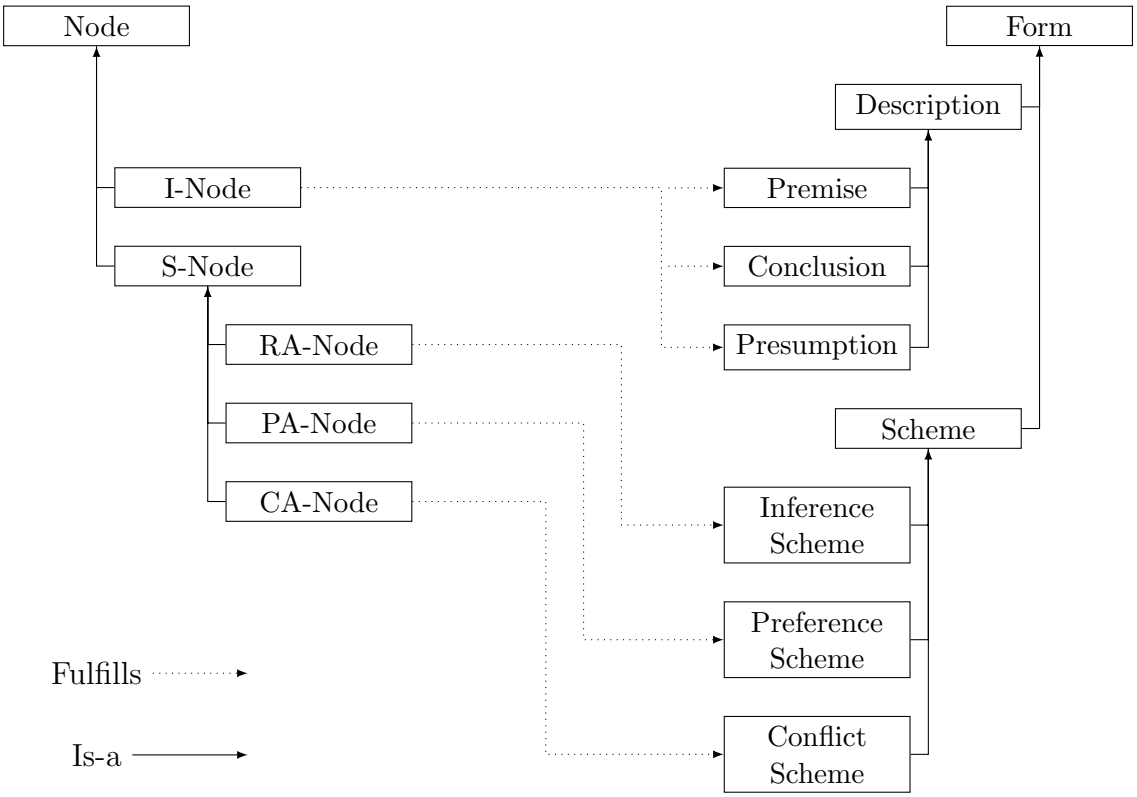


Figure 5.1: The core concepts in the AIF ontology

In figure 5.1, these two ontologies are displayed along with the possible 'fulfills'-relations between them. The edges in the upper layer and the uses-relations in the forms layer are not included in this diagram.

5.1.2 AIF+: extension for dialogue

Although there is an elaborated description of the concepts in argument networks and the forms they fulfill, the communicative aspect of argumentation is quite underexposed in the AIF description in [18]. The concepts of asserting and questioning a proposition are missing from the current ontologies. Neither exists a way to express whether a premise or conclusion is left implicit (not asserted).

In [20], a correspondence between the AIF and a so-called pragma-dialectical approach is established. It is based on the AIF+, which is an extension by [14] to connect dialogue obeying certain protocols to argument networks. The IAF+ considers locutions to be a special kind of I-nodes called L-nodes. The proceeding of the discourse is expressed in *transitions*, which are applications of *illucutionary schemes*. These schemes represent the dialogue protocols from pragma-dialectics and dialogue games.

This extension is not suited for the ArgAF however, for the following reasons:

- Natural argumentation, even in a formal setting, is not bound to a finite set of dialogue protocols, or at least there is no extensive description of the way transitions may occur in

natural discourse. The normative approach to argumentative discourse is thus too narrow for the ArgAF.

- Recall from chapter 3 that the ArgAF requires a distinction between the dialogue domain (containing locutions and speakers, which can be annotated directly) and the argumentation domain (which contains an *interpretation* of the locutions to be arguments). The AIF+ incorporates the communication concepts in the same layer as the argumentation, which obscures this distinction.

- As the documents the ArgAF is primarily concerned with do not represent the discourse verbatim, one cannot expect to be able to recover all transitions from the document. For instance, the order of the locutions might not be the same in the document as was it in the real discourse. It is better to not make assumptions about the discourse if there is no need to do so.

5.2 GrAF: Graph Annotation Format

Recall that the annotation requirements were based on the notions from the LAF (2.1). A concrete extension for the LAF is called GrAF and is described in [8]. It views annotations as directed graphs, rather than trees. This means an annotation can span both segments of the primary data and other annotations at the same time. There are two kinds of nodes: segment nodes and annotation nodes. The edges constitute the *referential structure*. The*feature structure representation* composes the content information, which is done by *labeling* edges and annotation nodes.

The primary data has a virtual *sink* node between each pair of characters, edges connecting these are proper segments. The resulting graph G serves as a basis for the annotation graph: the edges in G are treated as nodes ('segments') in the final annotation graph. In practice, this means a *segment* is a node that stores two character offsets. Segment nodes have no content information (labels).

Annotation nodes are labeled with one or more features, which have a name and a value. A node can have an arbitrary number of such labels and there is no constraint on the features, such as a the existence of a main type (classification). The GrAF does not specify *content categories*, annotation types, but refers to the *Data Category Registry* (DCR) that is part of the LAF architecture. Hence, it is possible to specify schemas that annotations should fulfill. Note that this possibility to externally specify schemes resembles the forms layer in the AIF ontology.

Annotation nodes reference segment nodes, other annotation nodes, or both. The referencing is performed by directed edges, which can be labeled with features as well. The resulting annotation graph is not allowed to have cycles and all paths that start in annotations, need to end in segments.

The GrAF structure is illustrated in figure 5.2. An annotation graph is displayed with annotation nodes as rectangles with rounded corners (the node text shows its labels), edges as (possibly labeled) arrows, and segment nodes as small rectangles. The segments reference characters in the example sentence, while in fact a segment just stores character offsets of course.

The figure illustrates that there may be several paths from one annotation down to segments: the sentence annotation references both a segment and its constituents. It also shows how one segment can have several annotations: the segment 'Robin' is referenced by different annotation nodes. The labeling of edges is optional, indeed not all the edges are labeled in the figure.

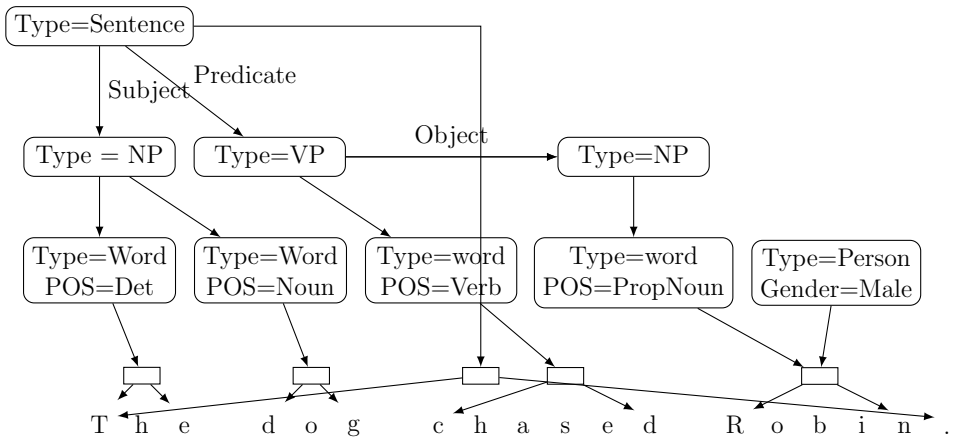


Figure 5.2: An example annotation graph

5.3 A model for the ArgAF

In figure 5.3, the model for the ArgAF is presented in an ER diagram. The entities are presented as rectangles, relations as diamonds, and attributes as ellipses.

The diagram consists of four parts: the forms layer and the upper layer, which are adopted from the AIF ontology; the annotation schemes, which describe how annotations can be created; and the objects layer, in which the actual annotations reside.

The upper and forms layers The ArgAF incorporates the concepts of the extended AIF model that was described in 5.1.1, as it is considered to be a solid foundation that connects to many argument processing tools. Not that three modifications are made to the model.

Firstly, the subtypes of the S-Nodes (RA, PA, CA) are not displayed as distinct entities in figure 5.3, but instead the kind is added as an attribute. The same goes for schemes. This is because in the ArgAF, the types only function to make sure an S-Node of some kind fulfills a scheme of the corresponding kind. There is thus no need to model them as different entities.

Secondly, the modeling of critical questions as either assumptions or exceptions is questionable: exceptions are assumed to be negated assumptions and vice versa. The assumptions are therefore discarded as a separate category. Just like in the AIF ontology, exceptions are premises to conflict schemes that are used by inference schemes.

Thirdly, a more conceptual modification is made: the descriptors are considered to be just one type. The usage relation (scheme uses descriptor) is assigned the role of the descriptor: premise or conclusion (recall that assumptions were already discarded). This adjustment saves space, as similar descriptors only need to be saved once. Moreover, it seems to be a more accurate model of the situation. The sole fact that it is *used* by an inference causes a descriptor to be a premise or conclusion.

Note that there is no information about locutions or other discourse aspects in the upper layer, only the argument$_1$ structure and semantics are stored here.

The object and annotation schemes layers For the annotations, the ArgAF incorporates a slightly altered version of the GrAF model from section 5.2. The adjustment lies in the fact that the ArgAF model requires each annotation to have a type that is defined in the annotation scheme layer. Furthermore, mandatory *referents* can be defined in that layer as is explained below. On the other hand, assigning features to annotations is not mandatory and entirely takes place in the object layer.

Observe that the division in an object and a scheme layer resembles the division in the other two layers: in the one part it is specified how the structure can be established, in the other part

are the instantiations. Take a look at the annotation schemes layer. The annotation types should encompass at least locutions, and in particular assertions. The main focus is the annotations of discourse concepts, but there are no restrictions on the addition of other annotation types, such as linguistic entities.

As is the case in the GrAF model, annotations may reference both text segments and other annotations. In the ArgAF model, the edges from an annotation to its references are labeled as well. This accommodates an extended notion of 'feature': an annotated piece of text does not 'accidentally' reference segments or other annotations. Rather, this referencing has an explicit semantic sense. The need to refer is included in the Annotation Scheme Layer, which contains an entity 'Referent' to express that there must be something to refer to. This entity has no additional features or role.

For instance, each locution must reference a speaker (which can be a 'named entity' annotation or a text segment), a verb (which expresses the so-called illocutionary force of the locution: a verb like 'maintains' expresses another kind of assertion than 'admits'). And for certain, it needs to explicitly reference its content as either other annotations (such as statements) or a text segment (or perhaps both).

Connecting the layers The upper layer is connected to the forms layer via the familiar 'fulfills'-edges. Although the fulfillment primarily links the entities, such as I-Node to Descriptor, the relation could be emphasized to hold between the Edges (in the Upper Layer) and the Uses-edges (in the forms layer) as well. Indeed, the latter seems a better representation as one I-Node may play a role in different arguments, and could thereby fulfill several Descriptors. The edges in the Upper Layer have a unique role however.

The fulfillment of annotations embodies the fact that each annotation needs to have a type. The Referent can be fulfilled by both text segments and other annotations, but each 'Needs'-edge should be matched by a reference in the object layer.

The connection between the object layer and the upper layer, which is the central issue of the ArgAF, is carried out by the 'Expresses'-relation between Annotation and I-Node. In particular, a specific annotation can be part of this relation, namely a Statement. A further requirement could be posed that a statement can only link to an I-node if it is referenced as content by an assertion.

The text may very well contain words or phrases that indicate the presence of argumentation, such as "therefore". It might seem desirable to link the annotation of such a phrase to an S-Node. This is not allowed in the ArgAF model. The rationale is that the inference step does not belong to a particular text segment, but rather emerges from the concurrence of statements. Note however that it is permitted, and even recommended, to annotate such phrases. These annotations could indeed help the annotator (and accordingly the machine learner) to detect the argument.

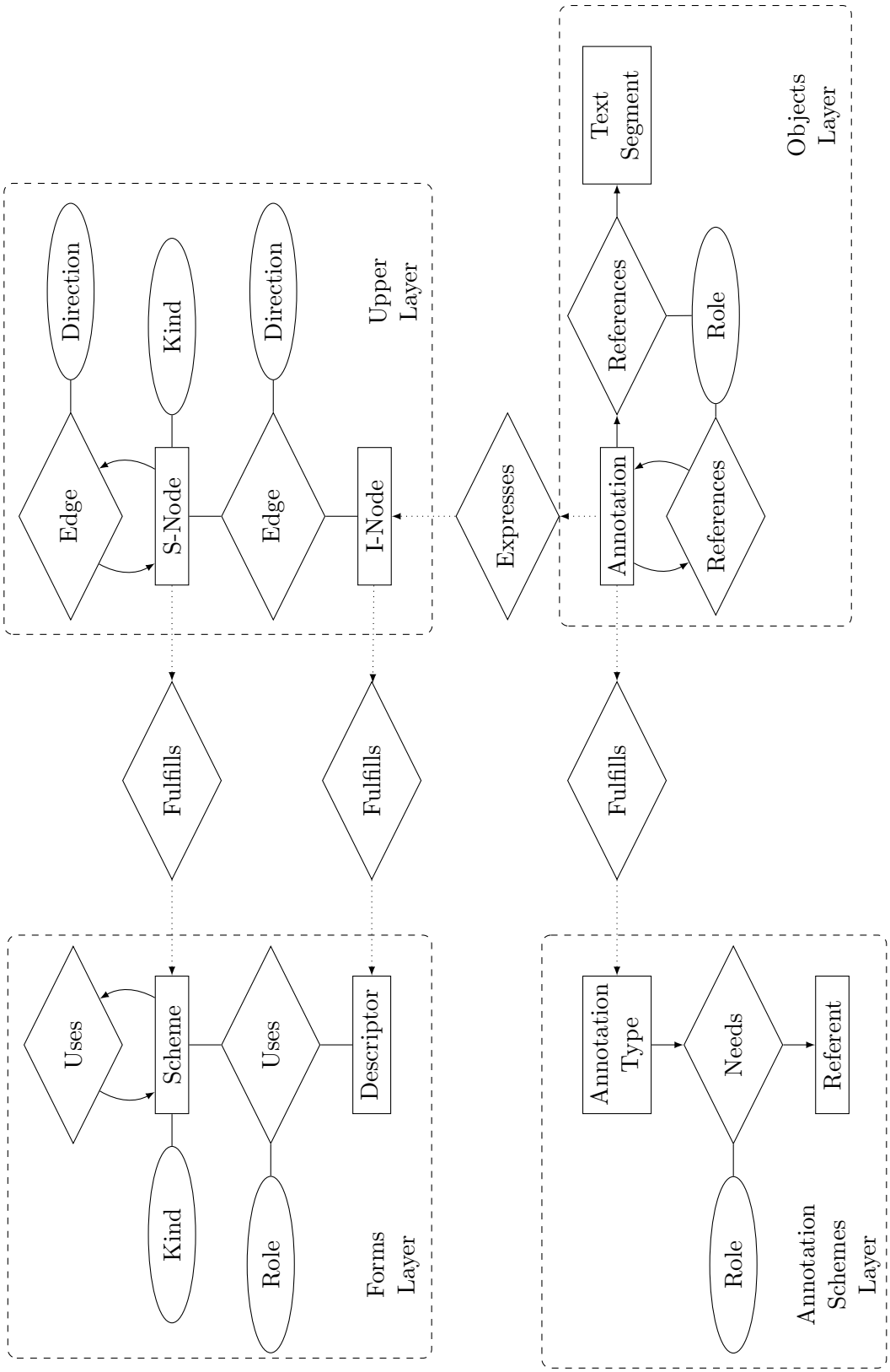


Figure 5.3: The ArgAF model in an ER diagram

Chapter 6

Specification of the Proposed Framework

The data model for the ArgAF that was described in chapter 5 can serve as a basis to specify the data format. The two parts of the framework, corresponding to the annotations themselves (the product) and the process of annotation, are described in the following sections. After this, a description is given of how the framework fulfills the requirements from chapter 3.

6.1 Data format

According to the requirements, the annotations should be stored in a 'rigid dump format' that is isomorphic to the data model, which was described in section 5.3. A relational database is considered a good representation of the annotations. A relational database is a durable way to store information and allows for flexible querying of the data, so that tools may access the information in different ways. For example, a machine learner may ask for any annotations that cover some part of the textual data.

The database design is called ArgAFdb. A more concrete implementation (in SQL for instance) can follow this description. It was inspired by a database definition called AIFdb that was implemented in Dundee for the successor of AraucariaDB, which is displayed in figure 6.1 and was described in [21]. However, there are some differences between the two designs, which are described below.

The ArgAFdb is divided into layers, corresponding to four layers from the data model and a fifth layer that is added to contain meta-information. Tables can reference a table in another layer; the division exists for the purpose of clarity only. For each layer, a figure is presented depicting its tables. The key attribute(s) are underlined, foreign keys are preceded by a symbol: $\ll$. A description is given below of the structure of the layers and the design decisions, based on the AIFdb.

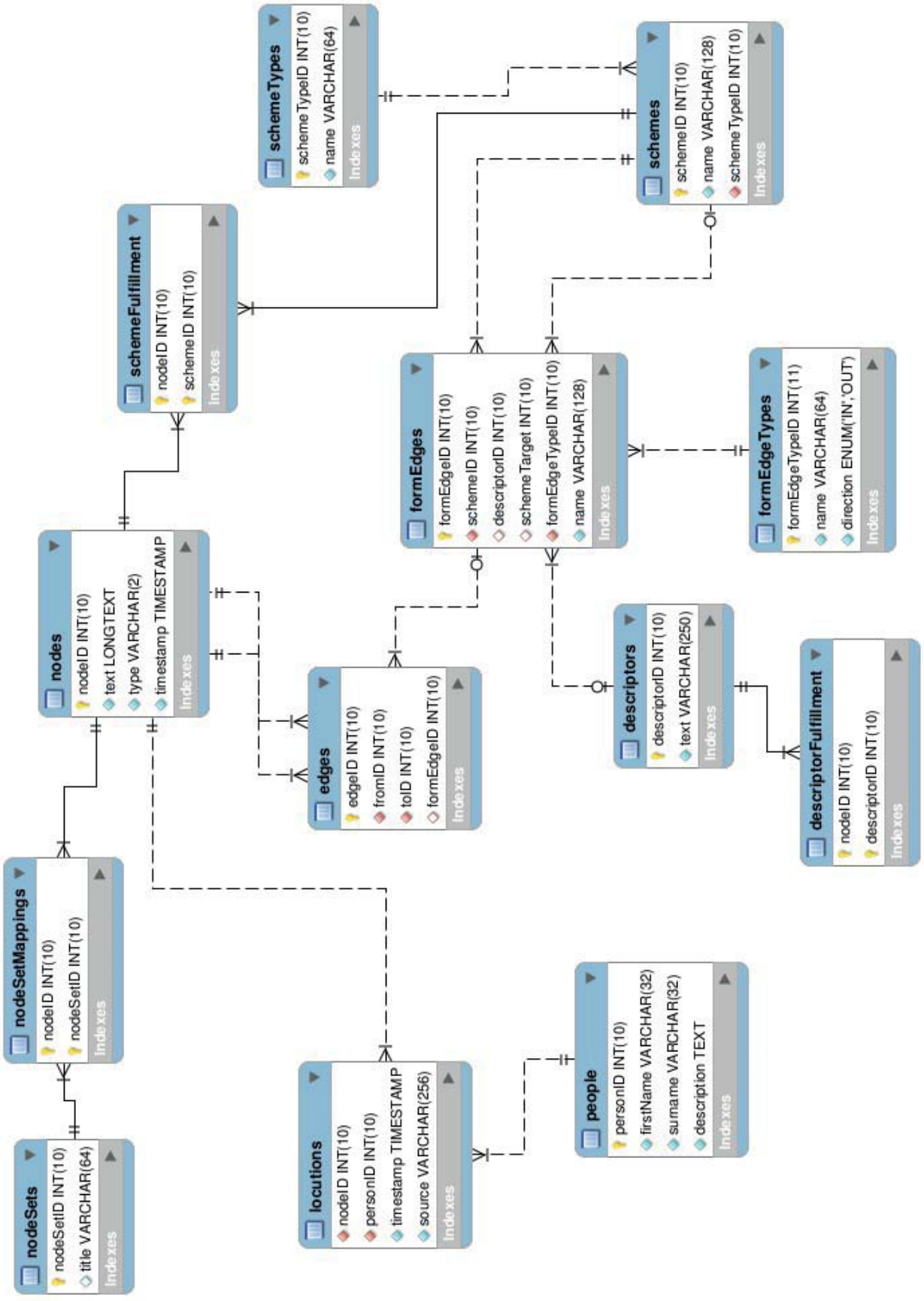


Figure 6.1: AIFdb, the AIF-based Database definition by Dundee for the Araucaria project that was used as a starting point for ArgAFdb

- **The meta layer** (figure 6.2) contains information about the annotation process.

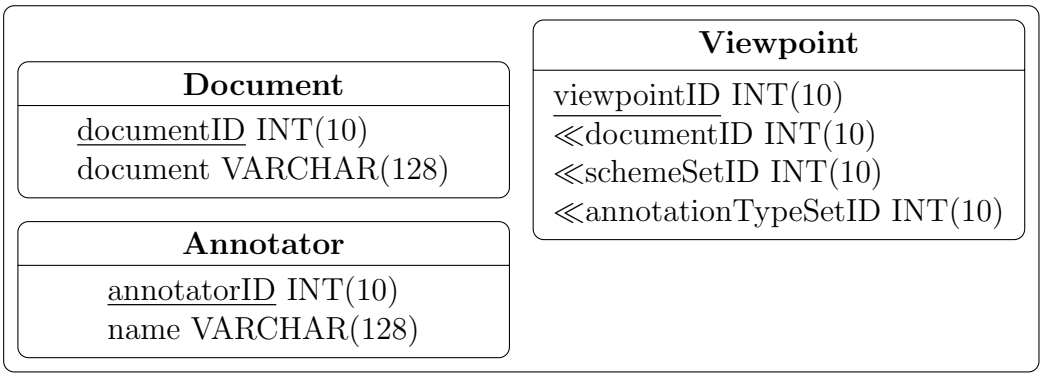


Figure 6.2: ArgAFdb - The meta information

The administration of documents is left outside of this database for the user to implement. The attribute `document` is designed to store a reference to external information about the document including its location in the file system and its source, among others. The table `Annotator` stores the name of the annotator, any additional information needs to be stored outside of the database.

The table `Viewpoint` keeps track of the document as well as the scheme set and the annotation type set. Each new tuple in the object layer or upper layer references such a viewpoint, which expresses the theory the annotation was based on. The idea is that an annotator selects the scheme sets he will use before starting to annotate a document. A further description of the scheme set and annotation type set is given in their respective layers.

- **The forms layer** (figure 6.3) contains the two types of forms and the *use*-edges connecting them.

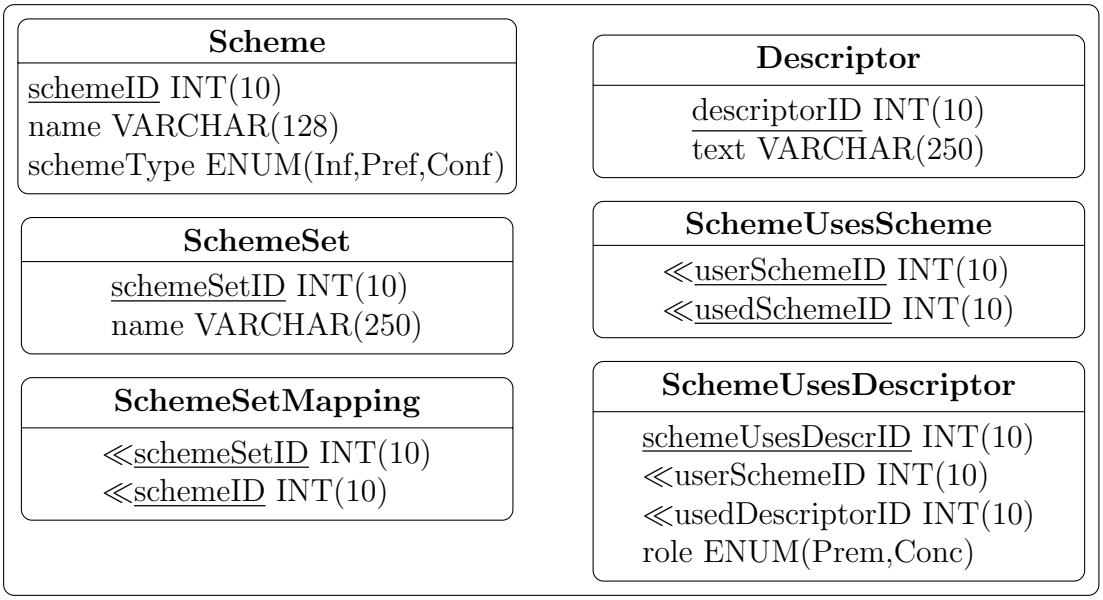


Figure 6.3: ArgAFdb - The forms layer

In both AIFdb and ArgAFdb, schemes and descriptors are expressed in distinct tables. In the AIFdb there is only one `formEdges` table, with a separate table to store the semantics of the form edges. In the ArgAFdb, there are two tables for form edges: a scheme may use either another scheme (such as a conflict scheme specifying an exception to an inference

scheme), or a descriptor (a premise or conclusion belonging to an inference scheme). Recall from section 5.3 that the role of the descriptor is modeled in this relation, rather than storing it as a feature of the descriptor itself.

The separate table `schemeTypes` is not adopted by the ArgAFdb, instead the possible scheme types are enumerated.

The table `SchemeSet` enables the definition of several scheme sets that can be chosen by an annotator to use. This allows for flexibility as new scheme sets may be defined in the future. The `SchemeSetMapping` is a separate table so that the same scheme may be part of several scheme sets. A scheme set can be easily extended because of this. A descriptor can not exist on its own: it is always used by a scheme, thus there is no need to map the descriptors in the scheme set.

- **The annotation schemes layer** (figure 6.4) stores the annotation types and the way they should reference.

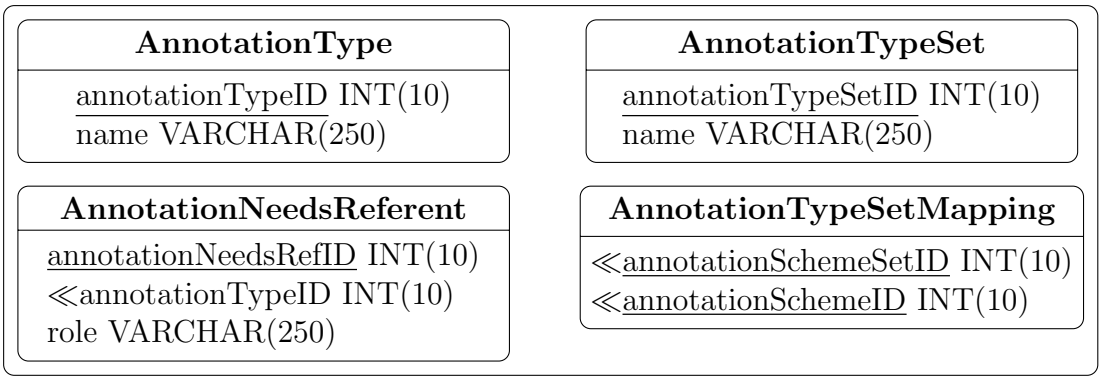


Figure 6.4: ArgAFdb - The annotation schemes layer

The AIFdb allows for speech acts to be stored in a table `locution`, referencing a person (the speaker), a time stamp (the time of the actual utterance) and a source (URL) of the locution. The time stamp may be hard to instantiate, especially for locutions that were found on the internet, so one might assume that either one of these values be `NULL`.

The ArgAF has a richer structure to store not only locutions but other annotations as well. The annotation schemes layer has a table `AnnotationType` that should contain at least the type *statement* and probably *locution*, but other annotations can be added as desired. The role of the `AnnotationTypeSet` is comparable to that of `SchemeSet` in the forms layer.

For each annotation, it can be specified what referents it needs in `AnnotationNeeds-Referent`. Recall that a loctution needs to reference a speaker, a verb and its content. The role is added as an attribute to this table.

- **The upper layer** (figure 6.5) contains nodes and edges between nodes.

  The AIFdb contains a single table for both S- and I-Nodes. As a result, only one table is needed to store the edges in the upper layer. A different table `nodeSetMappings` stores the type of the nodes. In the ArgAFdb there are separate tables for I- and S-Nodes, because they have different attributes. This creates the necessity to have three distinct tables for edges in the upper layer. This is not a problem however, because there are no references to edges in other tables. Remember that all three edge tables and the two node tables need to be consulted to obtain the complete argument network.

  Each S-Node fulfills exactly one scheme, so `SchemeID` is a foreign key in the table `SNode`. When a simple argument structure is instantiated based on a piece of text, an RA-Node (inference application) is created and it is mandatory to assign a scheme right away (which

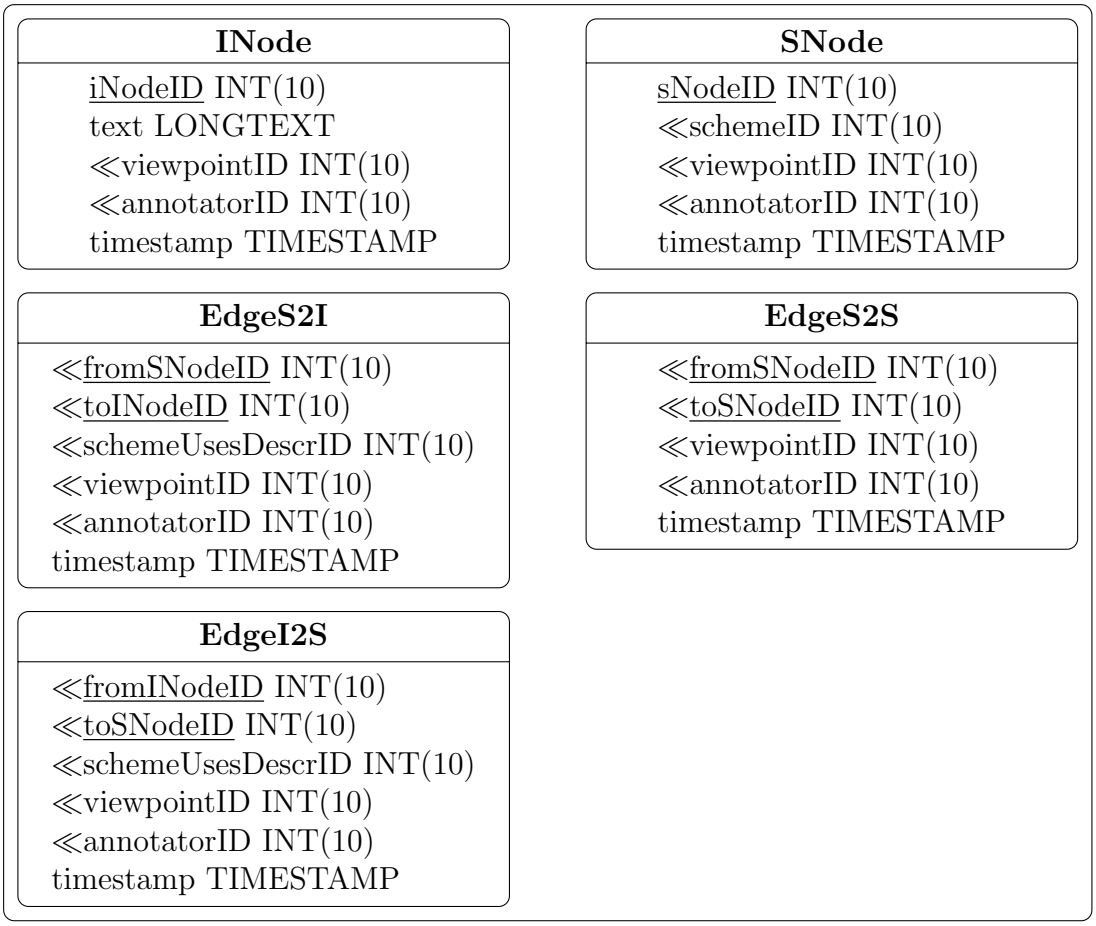


Figure 6.5: ArgAFdb - The upper layer

is to say, the feature `schemeID` has a `NOT NULL` constraint). All the descriptors that come with a scheme are required to be fulfilled by an I-Node. Whether or not that I-Node is expressed in a statement (an annotated piece of text in the primary document) reflects its enthymematic status. Missing premises can thus be attacked, as they do exist in the argument structure.

An I-Node may however fulfill several descriptors, belonging to the schemes of the supporting and supported S-Nodes. This is why the foreign key `SchemeUsesDescrID` is added to the tables `EdgeS2I` and `EdgeI2S`. The `text` attribute of the I-Node contains the proposition it expresses in natural language, although another (logical) language could be used as well.

All the tables store information about their creation: the viewpoint, the annotator, and a time stamp.

- **The object layer** (figure 6.6) contains annotations that are connected to the upper ontology.

  The most basic structural units are tuples in the table `TextSegment`. The start- and end-offset are stored, together with a reference to the document. This combination is required to be unique, there is no point in storing the same text segment several times.

  An `Annotation` has a type, which is a foreign key referencing an `AnnotationType` from the annotation schemes layer. Furthermore, it references other annotations or text segments via the tables `AnnotationRefAn` and `AnnotationRefSeg`. The meaning of the reference is derived from the `annotationNeedsRefID`.

  Recall that the GrAF puts some restrictions on the paths in the annotation graph. These should be translated to constraints and triggers for the actual database implementation.

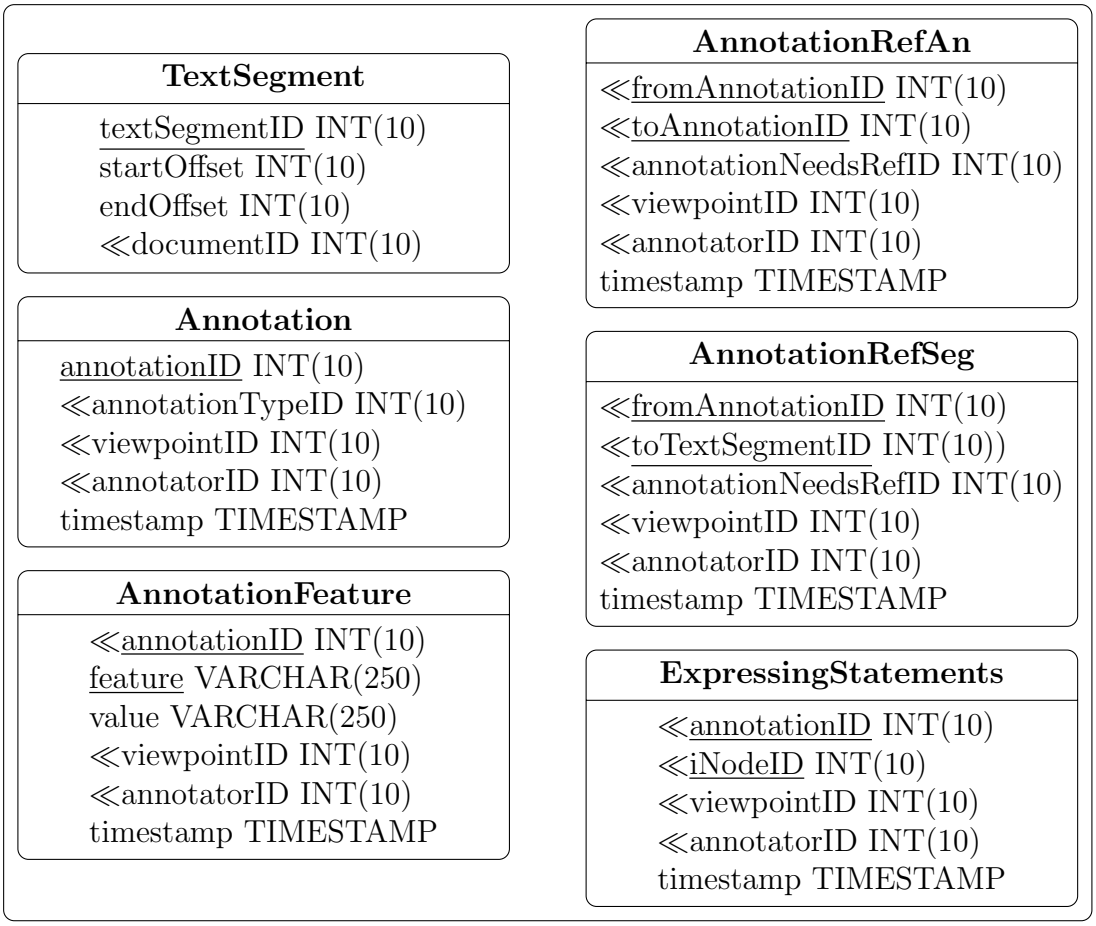


Figure 6.6: ArgAFdb - The object layer

The table `ExpressingStatements` links the annotations to the argument structure. A requirement of the `annotationID` attribute is that its referenced annotation be a *statement*.

Again, each table stores information about its creation.

6.2 The annotations

This section summarizes what the product of the annotation process is. The database design that was introduced in section 6.1 is the format to store it. The result is a set of actual annotations (linguistic and discourse) and an argument structure that is related to the text via the annotations. Therefore, content information concerns both the annotations and the argument structure whereas the structural information refers to the annotations only.

Content Even after the implementation of a concrete database, it would not be ready for annotating yet. Both the forms layer and the annotation schemes layer need to be instantiated, because the tables in the object and upper layer reference it. It is beyond the scope of this research to instantiate a complete set, but an example database is given in appendix A.

Scheme sets can be adjusted for several reasons. For one, the application of scheme sets in practice could reveal unwanted limitations. And should the theoretical basis for argument schemes develop, the ArgAF can still be used with an altered scheme set. Furthermore, a scheme set can be instantiated for a particular type of documents, for instance a scheme set containing a specialization of a particular argument scheme. Different annotators could also use a scheme set that suits their specialization. It is even imaginable to have annotators specify new (specializations of) argument schemes during the annotation process. Also annotation type sets could be changed,

for similar reasons. However, an uncontrolled growth of argument schemes and annotation types should be avoided. Indeed, it is harder to compare annotations from divergent viewpoints that come with different classifications.

After the forms and annotation schemes layer are instantiated, annotations can be created in the object layer that reference the annotation types they fulfill. Furthermore, the argument structure can be instantiated with nodes and edges referencing the tables in the forms layer and statements referencing I-Nodes.

Structural information The way annotations reference text segments or other annotations is included in the database. The document can be accessed via the table `Viewpoint`. How the file system containing the documents should be equipped and how the references point to the right files is deliberately left unspecified. It is up to the end users to decide on an appropriate representation of the documents. Remember that the documents are required to contain only Unicode characters to make sure their byte offset can be mapped to the character offset.

6.3 Procedural aspects

To be able to actually use the framework for annotating documents, a user interface needs to be implemented. Furthermore, a description is needed of the task and subtasks of the annotation.

User interface To accommodate for annotators, a user interface should be created that takes care of inserting entries in the database so the annotators would not need to worry about technical details. A user interface can additionally ease the annotating process, for instance by allowing annotators to select a piece of text for an annotation. The machinery of the user interface would translate this action to the internally represented `textFragment` with the correct byte offsets. Other helpful features would include the suggestion of all the required I-Nodes at the time an S-Node is created.

Unfortunately, the creation of even a minimal user interface was not achievable within this research project. This raises the entry threshold to actually use the framework, as a lot of work needs to be done before the annotators can be put to work. Existing environments can possibly be altered to facilitate the framework, or used for a part of the annotating tasks.

Tasks This section suggests a procedure, but it is recommended to be adapted in consultation with the annotators to obtain the best practicable solution. Alternatively, a different approach could be taken to specify a completely new procedure. The database specification offers flexibility in this respect. Note that it is recommended to monitor the feasibility of comparison between different approaches and procedures at specifying alternatives.

The procedure suggested here divides the actual annotation task is divided in two stages: annotating the discourse structure, and building the argument network. It furthermore specifies some preparatory tasks

- Preparations:
  - Selecting annotators

    The annotation task is not a trivial one: it relies on a lot of interpretation. The profession of the annotators is expected to influence the result. The obvious choice is between argument experts, linguistic annotators or legal experts (in the current case of filed legal cases). Each group has its own specialization and corresponding approach: legal experts best understand the subtleties of legal language, linguistic annotators are best aware of the way text segments can be interrelated, argument experts are trained to recognize the argumentative structure.

It is recommended to combine the expertise of the diverse professions, thus to have a group of annotators with divergent backgrounds work on the same documents.

- Determine a viewpoint

  Remember that the `Viewpoint` specifies the argumentation scheme set and annotation type set. As has been argued earlier, the formulation of new sets should be restrained. It is best to carefully consider what schemes are desired for the annotation of a range of document, before starting the rest of the procedure.

- Lecture for general knowledge

  This item is inspired by the procedure from [5, page 66]. It is recommended to have a session with the involved annotators to introduce topics such as the goal of the annotation process, the nature of the documents, the annotation tasks, and the user interface.

- Stage 1:

  The annotators are first required to annotate the discourse structure, that is: the locutions. Wherever possible, the statements that make up the content of the locutions should be distinguished and annotated. When performing this task, the annotator learns what statements the argument structure includes.

  It might prove to be convenient for the annotators to have some linguistic annotations as a starting point, such as sentences and part-of-speech (POS) tags (classification of words). These can be automatically extracted. It is possible that (parts of) the rest of stage 1 can be executed automatically, now or in the future.

- Stage 2:

  Based on the statement annotations, argument structures can be created. Argument schemes need to be assigned to inferences. These are supposed to help identify the premises and conclusion by means of the descriptors. The S-Nodes and edges should directly be connected to the proper descriptor, and the I-Nodes should be connected to statements immediately, or left unconnected to express their enthymematic status.

  It is possible that, on closer inspection, new statements need to be annotated or existing statements taken apart. The user interface should allow for this, but it is is discouraged to make many changes to the annotations.

  It is proved to be hard to obtain complete arguments, as premises and conclusions can be far apart in the text [5]. Also complex structures involving several arguments are hard to compose. Providing a graphical representation of the argument network might help the annotator keep an overview.

- Follow-up:

  After the group of selected annotators have performed stages 1 and 2, their results would hopefully have some essentials in common. Based on a comparison of their results, a new (and therefore somewhat unbiased) annotator should put together a gold standard. Because the argument structure is the core result, an argumentation expert is recommended to perform this last step.

6.4 How the requirements are met

Storage Both the annotations and the argument structure are stored in a relational database. This storage fulfills both the demands of flexible retrieval of the information and a stand-off

storage of the annotations. The database is based on the data model, thus the requirement of an isomorphic dump format is fulfilled.

Structural information Not only is the structural information kept separate from the documents (that is, the annotations are stand-off) but from the content too, based on the GrAF annotation structure. Annotation structures of arbitrary complexity can be created without problems but it is still possible to only consider some part of the structure. The structural information can be derived from the sub-annotations that can subsequently be discarded within some view.

Content information The data model is based on an existing model, the AIF, that describes argumentation$_1$ structures in such a way that they are translatable to other argumentation systems. All crucial concepts like statements, inferences and conflicts are recognized in the model. Argumentation schemes are incorporated in the model, but at the same time the option to append other argumentation theories is open. The argument$_2$ concepts are not included in the argumentation model itself, but can be annotated if requested.

Procedural aspects A suggestion is made for a procedure, which consists of two ordered stages. A general description of the tasks is given, a detailed version should be specified for actual usage.

Evaluation As has been said before, the post hoc evaluation can only take place after usage of the framework. The quality of the annotations can be trustworthy assessed after the framework has been used by several annotators to annotate a large amount of documents. The stability and reproducibility can be compared to the results that were achieved by other frameworks, such as the one described in chapter 4.

The usability of the resulting annotations for machine learning or argument processing can be assessed by performing those tasks with the annotations. The specific implementation of those tasks is quite divergent and also often it is based on the input material, which does not facilitate an easy comparison to annotations resulting from other frameworks.

Chapter 7

Conclusion

The focus of this thesis was to formulate the ArgAF: a framework for annotating the argument structure in filed legal cases, or generally documents containing resumé of discourse. To achieve this, the foundations of the fields of both argumentation and annotation were discussed in chapter 2. After this, the research questions that were set out in the introduction were answered in the subsequent chapters. In chapter 6 the resulting framework was presented as well.

This chapter summarizes how the research questions were answered. Next, the results are reviewed in the light of other research. Lastly, some suggestions for future research are made.

7.1 Answering the research questions

Recall the research questions that were presented in chapter 1:

(I.) *What requirements should be imposed on the framework?*

In chapter 3 the demands on storage, structural information, content information, and procedural aspects are put in place, based on the foundations that were investigated in chapter 2. The essential factor is to make clear distinctions between the different facets of the project. The evaluation techniques that can only be applied after the framework has been used, are introduced.

(II.) *What are some of the problems with the existing argumentation annotating framework that was proposed in [5]?*
The most striking problem is that the distinction between argument_1 and argument_2 is not reflected in the annotations. Entire sentences are classified as premises and conclusions, whereas they express discourse information as well. There is no way to designate the (smaller) statements, which are the building blocks of the arguments after all. The relations between arguments cannot always be derived, because there are no means to indicate that different sentences express the same (or even contradictory) statements.

(III.) *How can existing argumentation models be related to natural argumentation?*
Note that argumentation theories might be normative rather than descriptive.

(a) *What is a proper model of argumentation for this purpose?*
The argumentation schemes theory was chosen as a starting point, as it is an intuitive formalism that has been widely used and proved useful. In chapter 5 the AIF and the GrAF are introduced, which are abstract formats for argumentation and annotation respectively. It is argued that they provide concepts that constitute a good basis for the ArgAF.

(b) *What entities, such as propositions and attack relations, are defined in this model?*
Inference schemes specify which premises can yield which conclusions. These schemes provide semantics for an argumentation graph that consists of S-Nodes (expressing scheme application) and I-Nodes (expressing statements) and directed edges between them. The relations between arguments are not explicit in the model, but can be derived from conflict between statements.

(c) *How are these entities represented in written text?*
The distinction from chapter 2 between argument as a product (argument_1) and argument as a process (argument_2) reveals how statements in the text can be related to an argument_1 structure, whereas discourse aspects can be annotated but are not part of the argument_1 domain. Inferences themselves are not explicitly represented in written text.

(IV.) *How should annotations be stored?*

(a) *What annotation types should be defined?* The argument entities are not modeled as annotations but rather as nodes and edges in an argument network, which is a formal structure. Discourse elements can be annotated. There is not a fixed set of annotation types. At least a *statement* type needs to be defined, which can express the I-nodes in the argument network. It is recommended to define a *locution* type which references a speaker, verb, and content. Furthermore, annotating phrases that are indicative of argumentation could prove beneficial.

(b) *What features should be assigned to these types?* Annotating features of the discourse elements is not necessary, as long as the referencing obeys certain constraints. The entities in the argument network have semantics by fulfilling schematic descriptions, primarily argument schemes (introduced in chapter 2, but other theoretic approaches can be modeled as well.

(c) *What data format is suited for the storage?* In chapter 6, a relational database design is presented to store the annotations, based on the data model from chapter 5. It allows for flexible data manipulation and retrieval. However, its structure is quite complicated and not suited for direct human operation. A user interface still needs to be implemented to overcome this issue.

7.2 Implications towards the field of Artificial Intelligence

The development of systems that are capable of processing all kinds of documents is an interesting branch of AI. Especially extracting argument structures from textual documents is worthy of pursuit, as this has been proven to be a hard task even for humans [5]. The availability of corpora is a prerequisite for training machine learners. The only mature existing argumentation corpora have some limitations. The Araucaria project[1] has a database of annotated arguments. Only snippets of text are annotated though, while the context can be indicative of the kind of argument and thus should be used to classify argumentation. Moreover, argument detection can only be trained on arguments within their context. Mochales [5] has created a corpus of legal cases with annotations of the argument structure. However, the annotation framework she used showed some conceptual imperfections.

This thesis resulted in an annotation framework that resembles the more elaborated data model that underlies the Araucaria database, with an extension to cover complete documents. Furthermore, text fragments that strictly do not contribute to the argument structure itself, can be annotated and used for the tasks of argument detection and classification.

Although the ArgAF in its current form is not yet ready for usage, it is intended to provide a solid basis for the development of practical annotating applications. In this manner, this thesis has contributed to development of corpora that eventually can be used to train machine learners on the difficult task of detecting and classifying argumentation.

[1] http://www.arg.dundee.ac.uk/

7.3 Future research

Two directions for future studies are suggested, one concerning the evaluation of the framework as it is proposed in this document, the other working on extensions or improvements.

7.3.1 Evaluation

Annotating Recall from chapter 6 that the current framework is not ready for annotating yet. Some work needs to be done before the framework can be used and evaluated.

First, the database design needs to be implemented (in SQL, for instance). Then, a user interface needs to be created or an existing interface modified. For instance, a combination of an annotating interface like GATE [2]and the existing user interface of Araucaria. Finally, the procedural description needs to be elaborated.

Performance measures How the reproducibility and stability can be assessed is described in [15]. These quality measures were already computed for the framework by Mochales [5], so a comparison can be made. Computing these measures for Araucaria's AIFdb would determine ArgAF's position once more.

The convenience of the ArgAF for the annotating task is less easy to measure, but should add to the valuation of the framework. This is expected to depend greatly on the user interface.

Machine Learning The usability of the framework for machine learning of argument detection and/ or classification is the next goal. There is not yet a definite description of such a task, but a comparison of the performance would be an option if it were based on existing research. In [5] the experiments were focused first on the distinction between argumentative and non-argumentative sentences, and later between premises and conclusion. Another series of experiments aimed to parse documents and detect the complete argument tree-structure.

In [5] the approach was taken to view argument detection as a classification problem. Many machine learning algorithms exist for such tasks. The identification of the argument structure was performed with a rule-based parser. One new approach that could be taken in the machine learning is to use Markov Models, to create a probabilistic predictor based on the sequence of annotations that are encountered.

Argument processing The ArgAF was developed with a task in mind that has thus far been described as 'argument processing'. There is of yet no clear task description, but it would involve applications such as decision making based on argument evaluation, reasoning agents that add the arguments to their knowledge base, and perhaps agents that learn from historical arguments.

7.3.2 Improving and extending the framework

A critical discussion of the framework could reveal room for improvement. Some suggestions are presented here, along with some options for extension of the framework.

Theoretical considerations Some assumptions were made in the specification of the ArgAF. Other decisions could have been made of course, the effect of which could be the subject of investigation. The assumptions that are open for discussion include several topics.

One is the choice for argumentation schemes as a leading theory. There is room in the data model for alternative approaches, but there could be theories that are incompatible with the current framework. The flexibility of the framework in this sense could be studied.

[2]http://gate.ac.uk/

Also more generally, the role of (critical) questions in argumentation is an interesting topic. Is the assumption correct that questions do not reside in the argumentation$_1$ domain? What is the effect of a question on the argument structure? Questions are sometimes disguised statements, so questions can have divergent illocutionary force.

Enthymemes too are assumed to belong to the argument$_2$ domain in the ArgAF. Is this a correct assumption? A set of premises is specified for each argument scheme, an argument is enthymematic if one (or the conclusion) is missing. How can the completeness of these sets be justified? What about the assumptions that are expressed in the ciritical questions?

Discourse The objective of the ArgAF was to model the argument$_1$ structure. An obvious direction for extension is to try and model the argument$_2$ structure, that is the argumentative discourse. A requisite for this approach is to have documents with a (practically) complete discourse structure, such as verbose transcripts of debates. This comes with new challenges such as modeling the stages in the discourse and the effect of moves in the dialogue on the status of the argument. The pragma-dialectical approach could provide notions to accomplish this, although the question remains whether it is possible to capture natural argumentation in a finite set of dialogue rules.

Other argumentation theories For the ArgAF, the argumentation schemes theory was chosen as a starting point for the model. However, the options to adopt other theoretic approaches were deliberately left open. The extent to which this is actually possible is an interesting point of investigation.

Also alternative versions of the argumentation schemes theory could be investigated. For instance, the possibility of creating a classification system of argument schemes, seeing schemes as subtypes of others. This idea is elaborated in [3, chapter 10]. This approach might require a modification of the forms layer in the data model, and correspondingly in the database.

Relations between assertions As suggested in 2.2, deriving the relation between assertions from their content would be an interesting extension. A conflict relation can be more sophisticated than the relation between a statement p and its classical negation $\neg p$, and this goes for agreement relations as well. To begin, [3, section 7.2] suggests different types of opposition that can be distinguished.

Procedure As was indicated before, the procedure might be subject of improvement by consulting the annotators of a pilot. Alternatively, another procedure for the annotating task can be established. For example, an intermediate comparison of the annotation results could be inserted after stage 1, resulting in a common basis for stage 2.

Bibliography

[1] T. Bench-Capon and P. E. Dunne, "Argumentation in artificial intelligence," *Artificial Intelligence*, vol. 171, pp. 619–641, 2007.

[2] C. Reed and G. Rowe, "Araucaria: software for argument analysis, diagramming and representation," *International Journal on Artificial Intelligence Tools*, vol. 13, pp. 961–979, 2004.

[3] D. Walton, C. Reed, and F. Macagno, *Argumentation Schemes*. Cambridge University Press, 2008.

[4] V. W. Feng and G. Hirst, "Classifying arguments by scheme," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 987–996, 2010.

[5] R. Mochales Palau, *Automatic detection and classification of argumentation in a legal case*. PhD thesis, Katholieke Universiteit Leuven, 2011.

[6] N. Ide, L. Romary, and E. de la Clergerie, "International standard for a linguistic annotation framework," in *Proceedings of the HLT-NAACL 2003 workshop on Software engineering and architecture of language technology systems*, vol. 8, pp. 25–30, Association for Computational Linguistics, 2003.

[7] N. Ide and L. Romary, "Towards international standards for language resources," in *Evaluation of Text and Speech Systems* (L. Dybkjæ r, H. Hemsen, and W. Minker, eds.), vol. 37 of *Text, Speech and Language Technology*, ch. 9, pp. 263–284, Springer Netherlands, 2007.

[8] N. Ide and K. Suderman, "GrAF: A graph-based format for linguistic annotations," in *LAW '07 Proceedings of the Linguistic Annotation Workshop*, pp. 1–8, 2007.

[9] E. Pianta and L. Bentivogli, "Annotating discontinuous structures in XML: the multiword case," in *Proceedings of LREC 2004 Workshop on XML-based Richly Annotated Corpora*, pp. 30–37, 2004.

[10] H. Prakken and G. Vreeswijk, "Logics for defeasible argumentation," in *Handbook of Philosophical Logic* (D. Gabbay and F. Guenthner, eds.), vol. 4, pp. 218–319, Kluwer Academic Publishers, second ed., 2002.

[11] H. Prakken, "AI & law, logic and argument Schemes," *Argumentation*, vol. 19, pp. 303–320, 2006.

[12] F. H. van Eemeren, *Crucial Concepts in Argumentation Theory*. Amsterdam University Press, 2001.

[13] F. Bex, H. Prakken, and C. Reed, "A formal analysis of the AIF in terms of the ASPIC framework," in *Computational Models of Argument: Proceedings of COMMA 2010*, pp. 99–110, 2010.

[14] C. Reed, S. Wells, K. Budzyńska, and J. Devereux, "Building arguments with argumentation: the role of illocutionary force in computational models of argument," in *Computational Models of Argument: Proceedings of COMMA 2010*, pp. 415–426, 2010.

[15] S. Teufel, J. Carletta, and M. Moens, "An annotation scheme for discourse-level argumentation in research articles," in *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, (Morristown, NJ, USA), p. 110, 1999.

[16] D. Walton, *Informal Logic: A Pragmatic Approach*. Cambridge University Press, 2008.

[17] C. Chesñevar, J. Mcginnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott, "Towards an argument interchange format," *The Knowledge Engineering Review*, vol. 21, p. 293, 2006.

[18] I. Rahwan and C. Reed, "The argument interchange format," in *Argumentation in Artificial Intelligence* (G. Simari and I. Rahwan, eds.), ch. 19, pp. 384–402, Springer US, 2009.

[19] I. Rahwan, F. Zablith, and C. Reed, "Laying the foundations for a World Wide Argument Web," *Artificial Intelligence*, vol. 171, pp. 897–921, 2007.

[20] J. Visser, F. Bex, C. Reed, and B. Garssen, "Correspondence between the pragma-dialectical discussion model and the argument interchange format," *Studies in Logic, Grammar and Rhetoric*, vol. 23, pp. 189–224, 2011.

[21] J. Lawrence, F. Bex, C. Reed, and M. Snaith, "Aifdb: Infrastructure for the argument web," in *Computational Models of Argument: Proceedings of COMMA 2012*, pp. 4–5, 2012.

Appendix A

Example Database

An example instantiation of the database is presented in this appendix, along with a working example. All superfluous zeroes in identiefiers were omitted. The appendix comprises of the following figures and tables:

- In table A.1 the forms layer is instantiated. Minimal instances of the tables are given, with Argument from position to know (figure A.1) and Argument from memory (figureA.2). The mapping to the scheme sets is omitted here.
- In table A.2, the annotation schemes layer is instantiated. Only locutions and statements are included. The mapping to scheme sets is omitted again.
- A text fragment is presented in figure A.3 with a dialogue containing a classical example of the argument from position to know. The same text with character offsets displayed is also given.

  Although the fragment is much less complicated in language and structure, it resembles the documents the ArgAF is designed for as it is a resumé of discourse.
- In table A.3, the object layer is instantiated. The meta information (viewpoint and timestamp) are omitted. The information in italics and between brackets is provided for clarity, it is not stored in the database.
- In table A.4, the upper layer is instantiated. Again, the meta information is omitted.
- The corresponding argument structure is displayed in a diagram in figure A.4. The nodes are numbered to match the database instances. In the diagram, I-Nodes have two anchors for descriptors, corresponding to the incoming and outgoing edges.

Figure A.1: Argument from position to know, as described in [3, page 309]

ARGUMENT FROM POSITION TO KNOW

Major Premise: Source a is in position to know about things in a certain subject domain S containing proposition A.
Minor Premise: a asserts that A is true (false).
Conclusion: A is true (false)

Critical Questions

CQ1: Is a in a position to know whether A is true (false)?
CQ2: Is a an honest (trustworthy, reliable) source?
CQ3: Did a assert that A is true (false)?

Figure A.2: Argument from memory, as described in [3, page 346]

ARGUMENT FROM MEMORY

Premise 1: Persom P recalls ϕ.
Premise 2: Recalling ϕ is a prima facie reason to believe ϕ.
Conclusion: It is reasonable to believe ϕ

Critical Questions

CQ1: Was ϕ originally based on beliefs of which one is false?
CQ2: Is ϕ not originally believed for other reasons?
CQ3: Does the agent who recalls ϕ express doubt about ϕ?

Figure A.3: Example dialogue

Vera and Paul, who live in a small village in the south, had visited Amsterdam. They decided to return home and headed for the train station. Vera suggested to go right because she just asked a someone for directions. Paul argued those city people can't be trusted, they came from the left this morning.

```
[000]Vera and P[010]aul, who l[020]ive in a s
[030]mall villa[040]ge in the [050]south, had
[060] visited A[070]msterdam. [080]They decid
[090]ed to retu[100]rn home an[110]d headed f
[120]or the tra[130]in station[140]. Vera sug
[150]gested to [160]go right b[170]cause she
[180] just aske[190]d a someon[200]e for dire
[210]ctions. Pa[220]ul argued [230]those city
[240] people ca[250]n't be tru[260]sted, they
[270] came from[280] the left t[290]his morni
[300]ng.
```

Table A.1: Example database entries for the arguments in figures A.2 and A.1 (forms layer)

(a) Scheme

schemeID	name	schemeType
1	"argument from position to know"	"Inf"
2	"conflict from dishonesty"	"Conf"
3	"argument from memory"	"Inf"
4	"Conflicting statements"	"Cond"

(b) Descriptor

descriptorID	text
1	"Source a is in position to know about things in a certain subject domain S containing proposition A"
2	"a asserts that A is true"
3	"A is true"
4	"a is not an honest (trustworthy, reliable) source"
5	"Person P recalls ϕ"
6	"Recalling ϕ is a prima facie reason to believeϕ
7	"It is reasonable to believe ϕ
8	"B is true"

(c) SchemeUsesScheme

user-SchemeID	used-SchemeID
1	2

(d) SchemeUsesDescriptor

schemeUses-DescrID	user-SchemeID	used-DescriptorID	role
1	1	1	"Prem"
2	1	2	"Prem"
3	1	3	"Conc'
4	2	4	"Prem"
5	3	5	"Prem"
6	3	6	"Prem"
7	3	7	"Conc"
8	4	8	"Prem"
9	4	3	"Prem"

Table A.2: Example database entries for "Locution" and "Statement" (annotation schemes layer)

(a) AnnotationType

annotation-TypeID	name
1	"Statement"
2	"Locution"

(b) AnnotationNeedsReferent

annotation-NeedsRefID	annotation-TypeID	role
1	2	"Speaker"
2	2	"Verb"
3	2	"Content"

Table A.3: Example database entries for the dialogue in figure A.4 (object layer)

(a) Annotation

annotationID	annotationTypeID
1	2 (*Locution*)
2	1 (*Statement*)
3	1 (*Statement*)
4	2 (*Locution*)
5	1 (*Statement*)
6	1 (*Statement*)

(b) TextSegment

textSegmentID	startOffset	endOffset	
1	142	215	*(Vera suggested to go right because she just asked a someone for directions)*
2	142	145	*(Vera)*
3	147	155	*(suggested)*
4	157	167	*(to go right)*
5	176	215	*(she just asked a someone for directions)*
6	218	302	*(Paul argued those city people can't be trusted, they came from the left this morning)*
7	218	221	*(Paul)*
8	223	228	*(argued)*
9	230	263	*(those city people can't be trusted)*
10	266	302	*(they came from the left this morning)*

(c) AnnotationRefSeg

from-AnnotationID	toText-SegmentID	annotation-NeedsRefID
1	1	NULL
1	2	1 (*Speaker*)
1	3	2 (*Verb*)
2	4	NULL
3	5	NULL
4	6	NULL
4	7	1 (*Speaker*)
4	8	2 (*Verb*)
5	9	NULL
5	10	NULL

(d) AnnotationRefAn

from-AnnotationID	to-AnnotationID	annotation-NeedsRefID
1	2	3 (*Content*)
1	3	3 (*Content*)
4	5	3 (*Content*)
4	6	3 (*Content*)

(e) ExpressingStatements

annotationID	iNodeID
2	1
3	2
5	3
6	6

Table A.4: Example database entries for the dialogue in figure A.4 (upper layer)

iNodeID	text
1	To reach the train station, we should go right
2	A passer-by said the train station is rightwards
3	City people cannot be trusted
4	A passer-by knows the way in his town
5	To reach the train station, we should go left
6	Paul remembers that the train station was left
7	Paul could know, they came along this spot

(a) INode

sNodeID	schemeID
1	1
2	2
3	4
4	3

(b) SNode

from-SNodeID	toINodeID	schemeUses-DescrID
4	5	7
1	1	3

(c) EdgeS2I

from-SNodeID	toSNodeID
2	1

(d) EdgeS2S

fromINodeID	toSNodeID	schemeUses-DescrID
7	4	6
6	4	5
5	3	8
1	3	9
2	1	2
4	1	1
3	2	4

(e) EdgeI2S

Figure A.4: Example argument structure