Working on the Argument Pipeline: Through Flow Issues between Natural Language Argument, Instantiated Arguments, and Argumentation Frameworks

Adam Wyner[1] and **Tom van Engers**[2] and **Anthony Hunter** [3]

Abstract. In many domains of public discourse such as arguments about public policy, there is an abundance of knowledge to store, query, and reason with. To use this knowledge, we must address two key general problems: first, the problem of the knowledge acquisition bottleneck between forms in which the knowledge is usually expressed, e.g. natural language, and forms which can be automatically processed; second, reasoning with the uncertainties and inconsistencies of the knowledge. Given such complexities, it is labour and knowledge intensive to conduct policy consultations, where participants contribute statements to the policy discourse. Yet, from such a consultation, we want to derive policy positions, where each position is a set of consistent statements, but where positions may be mutually inconsistent. To address these problems and support policy-making consultations, we consider recent automated techniques in natural language processing, instantiating arguments, and reasoning with the arguments in argumentation frameworks. We discuss application and "bridge" issues between these techniques, outlining a pipeline of technologies whereby: expressions in a controlled natural language are parsed and translated into a logic (a literals and rules knowledge base), from which we generate instantiated arguments and their relationships using a logic-based formalism (an argument knowledge base), which is then input to an implemented argumentation framework that calculates extensions of arguments (an argument extensions knowledge base), and finally, we extract consistent sets of expressions (policy positions). The paper presents a progress report towards the implementation of dynamic, web-based, distributed, collaborative knowledge base construction in natural language which supports incomplete and inconsistent knowledge.

1 INTRODUCTION

In many domains of public discourse, participants have the opportunity to present their views. In public policy-making discussions, members of the public and politicians gather information and consult about issues of public import such as copyright, banking, immigration and others. Such consultations promote democratic institutions, support regulatory compliance, and make government more efficient. With the advent of the Internet, the opportunities to broaden such public participation have increased dramatically as has the volume of information. While consultation exercises are necessary and beneficial, the volume and complexity of the information introduces problems to store, analyse, represent, and reason. Moreover, the participants may enter uncertain, incomplete, or inconsistent statements; the knowledge may be non-monotonic, where statements are introduced or removed, context changes, and inferences change relatedly. These create substantive problems in reasoning.

Standard, current tools which are used to gather knowledge of policy are face-to-face meetings and rounds of consultation/commenting on written position reports. On line forums are also used to broaden participation and make policy more efficiently [23]. Broadly speaking, the meetings and on line forums serve to record the uncertain and potentially inconsistent points of view of the participants. Analysts subsequently try to identify and reason with the inconsistency in order to formulate and clarify the policy, a task which is done "manually". Moreover, the tools do not support participants to submit their contribution in a way that facilitates further processing.

None of the current tools address the knowledge acquisition bottleneck, nor knowledge representation and reasoning (KR) as understood in the subfield of artificial intelligence. The knowledge acquisition bottleneck is the problem of translating between the form in which participants know or express their knowledge and the formal representation of knowledge which a machine can process; the bottleneck has limited the advance of artificial intelligence technologies [18]. KR is understood as a structured, logical representation of the knowledge for automated reasoning, querying, and processing. Currently, knowledge engineers (or legislative assistants) manually structure and formalise unstructured linguistic information into a knowledge base (or legislative act) which supports information extraction, identification of relationships among statements, inference, query, development of ontologies, redundancy, identification of contradictions, reasoning with uncertain and inconsistent knowledge, or visualisations. Yet, in doing so, the knowledge engineers rely on their own fine-grained, unstructured, implicit linguistic knowledge, which itself is problematic.

In order to support policy-making, we address the knowledge acquisition bottleneck and inconsistency of statements with automated techniques in natural language processing, instantiating arguments, and reasoning with the arguments in argumentation frameworks. We discuss application and *bridge* issues between these techniques, outlining a pipeline of technologies whereby: expressions in a controlled natural language are automatically parsed and translated into a logic

[1] Corresponding author. University of Leeds, United Kingdom, email: adam@wyner.info.

[2] Leibniz Center for Law, The Netherlands, email: vanEngers@uva.nl

[3] University College London, United Kingdom, email: a.hunter@cs.ucl.ac.uk

(a literals and rules knowledge base – LRKB) using *Attempto Controlled English* (ACE)[4]; from this knowledge base, we generate instantiated arguments and their relationships using a logic-based formalism (an argument knowledge base – AKB) with as system based on [5]; we input this knowledge base into an implemented argumentation framework which calculates extensions of arguments (an argument extensions knowledge base AEKB) with [17] that is grounded in [14]. At the end, we extract consistent sets of expressions (policy positions).[5]

While each of these components is well developed independently, little consideration has been focused on the *integration* and *through flow* between them. The important novelty of this paper is an exploration of these bridge issues which must be addressed to construct an on line system which allows users to enter expressions in natural language that feed into an implemented logic-based formalism and argumentation frameworks, creating an *argument processing pipeline*. The objective of the pipeline is support and automate what is now an unsupported and manual task: input a (possibly inconsistent, dynamic) knowledge base of literals and rules expressed in natural language and output *policy positions*.

Many questions arise. Where we rely on input from public participants, who are not logicians or knowledge engineers with training in building well-formed rules, ill-formed arguments could be entered. What prompts can be introduced to make KB construction systematic and meaningful? Is Propositional or First-order logic appropriate as a semantic formalism for natural language. What limitations are imposed by using a controlled language? How can context and dynamics be addressed? In the face of these questions, our approach is pragmatic and incremental, using available technologies while acknowledging limitations. For example, we do not address the problem of ill-formed arguments; First-order logic is known to have a range of limitiations; controlled languages are inherently less expressive than 'full' natural languages, and we consider only *static* knowledge bases relative to a *fixed context*. These are abstractions which helps us focus on other central aspects of the system. Given further theoretical and technological advances, our approach is to be extended and refined towards the goal of an argumentation system which supports dynamic, web-based, distributed, collaborative knowledge base construction using natural language and reasoning with incomplete and inconsistent knowledge. This paper represents an intermediate report towards that goal.

In Section 2, we provide a sample policy-making discussion, a graphical representation of the relationships among the statements, and our translation methodology using the Attempto Controlled Language System. In this section, we develop a literals and rules knowledge base (LRKB), which represents the domain knowledge. In Section 3, we take LRKB and instantiate arguments for and against statements, forming an argument knowledge base (AKB). In Section 4, we take the arguments and relations defined in AKB as input to an argumentation framework AF and calculate extensions of arguments, which are sets of mutually compatible arguments, which forms an argument extensions knowledge base (AEKB). From the extensions, we extract the literals and rules for each extension to provide *policy positions* as shown in Section 5. The paper concludes with a discussion section.

[4] `http://attempto.ifi.uzh.ch/site/description/`

[5] The term policy position is perhaps related to *normative positions* of deontic logic [29].

2 Example, Translation Methodology, and Considerations

In this section, we discuss our source material and working example, the methodology of using the Attempto Controlled Language system to translate the sentences to parse and semantically translate the sentences, and considerations which arise in developing a set of sentences for input to an argumentation system.

2.1 Sources and Examples

[40] present a policy-making discussion adapted and abstracted from a BBC *Have Your Say* discussion list. The BBC's *Have Your Say* introduces a question for discussion, provides brief background points, then allows users to write in comments in a monitored, unthreaded discussion list.[6] For the particular study, [40] considered comments in response to the question *Should people be paid to recycle?*, where 697 comments were published (and 35 were rejected).[7]

The comments as they are present a knowledge engineer with a wealth of unstructured textual data since the discussion list is unthreaded and has very few substantive constraints on what participants can contribute or how. Among the many issues such a discussion raises for the analyst, we see the following. Participants may use any vocabulary or syntactic structure they wish (so long as it does not lead to the rejection of their comment); they may use novel words or expressions, metaphors, misspellings, and ungrammatical sentences. There is no check that the meaning as interpreted is the meaning as intended. Implicit or pragmatic information in the comments must be inferred by the reader. There are no over indications of inter-comment relationship (unless the commenter explicitly refers to one or more previous comments). The comments can represent uncertain or inconsistent information. The comments can be vague and ambiguous. The interpretation of comments is left to the analyst to infer.

Beyond these empirical observations, it is unclear what current computational linguistic system could accurately parse the large majority of sentences, yield the acceptable correlated semantic interpretation, and be used to support the acquisition of the knowledge. Textmining can extract some useful information (e.g. sentiment analysis [25], some aspects of rules [39], automatic implicatures [13], and contradictions [32]), but is still limited, error prone, and does not capture the complex meaning of each comment, much less the complex web of meanings comprised of the semantic relationships among the comments. Uncertainties and inconsistencies (other than explicit contradictions) are difficult to identify. It would be a significant task to take such a list of comments and transform them into a knowledge base that is suitable for a significant KR task.

[40] scope the issues, which is a common practice in semantic studies in Linguistics. By scoping the issues, the data is filtered, normalised, and reduced in scale so as to define a problem space that is on the one hand close to or derived from the source data and is on the other hand amenable to yet challenging for the theoretical and implemented techniques. It allows us to address and solve particular problems, abstract issues, and make progress which otherwise is prohibited by an ill-defined scope. While the resulting example is relatively far from the source material, it is still a useful starting point. The intention is that by addressing relatively artificial examples, we

[6] `http://news.bbc.co.uk/2/hi/talking_point/`

[7] See `http://newsforums.bbc.co.uk/nol/thread.jspa?forumID=7269&edition=2&ttl=20100218141845`.

can highlight issues, then extend the approach incrementally to solve additional and less artificial problems and examples.

With these programmatic points in mind, we adopt and adapt the approach and sentences in [40], where there are 16 sentences are automatically translated into First-order logic, manually indicating where there are rules and contradictions: users input sentences using the controlled language editor, then indicate *rule relationships* between sentences such as *premise*, *claim*, or *contradiction*. The result is a (possibly inconsistent) knowledge base.

In the current paper, we have 19 example sentences; in Section 2.3, we justify our revisions and discuss the sentences. For the purposes of this paper, we only work with Propositional Logic in this paper because the theory of argument formation for expressions in First-order logic is more complex than we need here and because the current implemented system for generating arguments and relations only works for Propositional Logic, though a system for First-order Logic is in development. A representation in First-order Logic is crucial in order to *identify* and *reason with* contradiction, redundancy, implication, ontological argument, fine-grained information extraction, and query; in future work, we anticipate working with First-order Logic Statements. We have, therefore, assigned each sentence a lower case propositional variable. Each rule has a lower case rule name.[8]

Example Sentences
p1: Every household should pay some tax for the households garbage.
p2: No household should pay some tax for the households garbage.
p3: Every household which pays some tax for the households garbage increases an amount of the households garbage which the household recycles.
p4: If a household increases an amount of the households garbage which the household recycles then the household benefits the households society.
p5: If a household pays a tax for the households garbage then the tax is unfair to the household.
p6: Every household should pay an equal portion of the sum of the tax for the households garbage.
p7: No household which receives a benefit which is paid by a council recycles the households garbage.
p8: Every household which does not receive a benefit which is paid by a council supports a household which receives a benefit which is paid by a council.
p9: Tom says that every household which recycles the households garbage reduces a need of a new dump which is for the garbage.
p10: Every household which reduces a need of a new dump benefits the households society.
p11: Tom is not an objective expert about recycling.
p12: Tom owns a company that recycles some garbage.
p13: Every person who owns a company that recycles some garbage earns some money from the garbage which is recycled.
p14: Every supermarket creates some garbage.
p15: Every supermarket should pay a tax for the garbage that the supermarket creates.
p16: Every tax which is for some garbage which the supermarket creates is passed by the supermarket onto a household.
p17: No supermarket should pay a tax for the garbage that the supermarket creates.
p18: Tom is an objective expert about recycling.
p19: If an objective expert says every household which recycles the households garbage reduces a need of a new dump which is for the garbage, then every household which recycles the households garbage reduces a need of a new dump which is for the garbage.

[8] One general issue that arises about the pipeline concerns the representations which suit each application. This is discussed further below.

As in [40], where a rule has more than one premise, the assumption is that the premises must conjunctively hold for the claim to hold. The rules are *definite clauses* - Horn clauses with exactly one positive litera. We assume that users construct rules from the propositions.

Rule Set
r1: $[p4 \wedge p3] \rightarrow p1$
r2: $[p9 \wedge p18 \wedge p19] \rightarrow p4$
r3: $p10 \rightarrow p3$
r4: $[p12 \wedge p13] \rightarrow p11$
r5: $[p5 \wedge p15] \rightarrow p2$
r6: $[p6 \wedge p7 \wedge p8] \rightarrow p5$
r7: $p14 \rightarrow p15$
r8: $p16 \rightarrow p17$

We can represent the structure of the literals and rules into a graphic where solid lines indicate a premise relation and a dashed line indicates a contradiction.

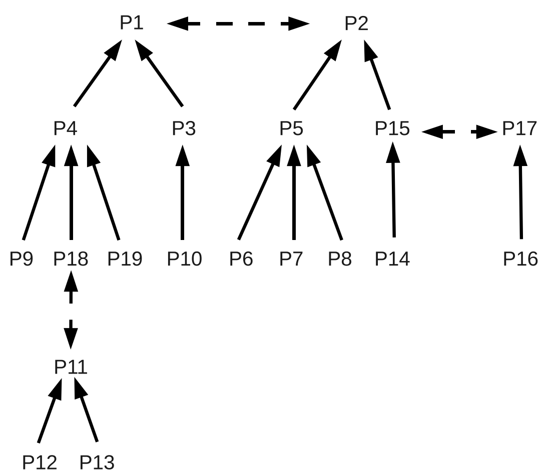

Figure 1. Relationships between Statements – Premises, Claims, Contradictions

In the next section, we briefly outline issues bearing on the syntactic parsing and semantic interpretation provided by *ACE*.

2.2 Parsing and Semantic Representation with Attempto Controlled English

[40] provide a detailed discussion of the application of Attempto Controlled English (ACE) to our candidate sentences. In the context of this paper, where the emphasis is on providing the relationship between the input statements, rules, and an argumentation system which works only on propositions, the importance of using ACE is reduced. Yet, it is relevant here to give a brief overview in that ACE

controls the language of the policy discussion and also provides expressions in First-order Logic that can be used in future implemented argument instantiation systems ([4] and [16]).

To facilitate the processing of sentences, we use a well-developed controlled natural language system – ACE (for an overview of the properties and prospects of controlled natural languages, see [37]). A controlled language has a specified vocabulary and a restricted range of grammatical constructions which are a subset of a natural language (e.g. English). Sentences written and read in the controlled language appear as normal sentences, but can be automatically translated into a formal representation.

ACE supports a large lexicon, a range of grammatical constructions, discourse anaphora, and correlated semantic interpretations: negation on nouns or verbs, conjunction, disjunction, conditionals, quantifiers, adjectives, relative clauses, discourse anaphora, modals (e.g. *necessity* and *possibility*), possessives, prepositional phrases, verbs with three arguments, and verbs with subordinate clauses. ACE checks that the sentences input to the system satisfy the constraints of the syntax and semantics of the language, thus the user is only able to input grammatically acceptable and semantically interpretable sentences in building the knowledge base. ACE has an associated reasoning engine to test for consistency and draw inferences.

As discussed in [40], the syntactic form must be crafted to generate the intended semantic interpretation, but this is relatively straightforward to do. Our 19 sentences are grammatically well-formed in ACE and have the intended interpretations.[9]

2.3 Considerations

In developing the sentences for input to a generator of instantiate arguments, three issues arose – making contradictoriness explicit, the construction of well-formed rules in the LRKB, and the introduction of implicit premises. We discuss these in turn.

To generate instances of arguments and their relations, we require explicit indicators of contradictoriness. To take a simple example, suppose that one participant makes a statement such as *Bill exercises daily* and another claims *Bill does not exercise daily*; the lexical items and syntactic form are such that it is relatively easy to identify these as contradictory. Systematically translating from natural language expressions with negation to logical forms with negation is a complex issue itself, though we do not discuss it here. However, rather than stating *Bill does not exercise daily*, suppose a participant claims *Bill is a couch potato*, which is also contradictory with *Bill exercises daily*. Yet, the lexical items and syntactic form are significantly different; it requires lexical semantic and syntactic knowledge to know that these expressions are contradictory. Indeed, there are a broad range of linguistic forms for expressing contradictoriness [11]; more broadly, contradictoriness is a rich, complex, poorly understood domain ([11], [9], [12, 32]). While *ACE* easily identifies the inconsistency between *Bill exercises daily* and claims *Bill does not exercise daily*, it misses the inconsistency between *Bill exercises daily*, and *Bill is a couch potato*.

There are two approaches to addressing such contradictions. One strategy is for the user to introduce an explicit *constraint* which simply states which two sentences are contradictory. If *Bill exercises daily* is represented as the propositional variable P, *Bill does not exercise daily* is $\neg$P, and *Bill is a couch potato* is R, $\neg$[P $\wedge$ R] is a constraint – P and R cannot both be true in a model. Yet, unlike the contradiction [P $\wedge$ $\neg$P] of Propositional Logic, constraints must be independently given in the knowledge base. Besides having to be specified, an additional problem of constraints is that there is no explanation of the contradiction; that is, they are *facts* of the knowledge base and follow from or relate to no other elements of the knowledge base.

To overcome this, an alternative strategy is to express the *enthymemes* or rules (if known) which give rise to the inconsistency; enthymemes are implicit premises that are required to allow an inference to go through and which must be made explicit in the KB. For instance, *Bill is a couch potato* implies that *Bill does not exercise daily*. Thus, in stating *Bill is a couch potato* one ought to introduce as well the rule that it implies that *Bill does not exercise daily*; then, where it is asserted that *Bill is a couch potato* and *Bill exercises daily*, we would find, by inference, that an inconsistency arises. Such an approach relies on some pre-existing and accepted terminology by the users, that is, that *exercise daily* is in the lexicon already such that the contradiction of it is meaningful.

A second issue relates to the construction of well-formed rules. Given a set sentences, participants can, in principle, introduce any combination of literals to produce a rule. Yet, some of these appear to be well-formed and meaningful, while others do not. For example, the following argument is formally and syntactically well-formed as well as semantically coherent: *Socrates is a man and Every man is mortal therefore Socrates is mortal.* In contrast, the following is formally and syntactically well-formed, but semantically incoherent, we would not want to accept it as an argument: *Bill is rich and Jill sails therefore Socrates is mortal.* Clearly in the latter example, we are missing some semantically relevant relationships between the premises themselves and between the premises taken together and the claim; in this regard, the former example is more coherent. While the former example may have a relatively simple explanation for this coherence (terms are shared), the coherence of more complex examples such as argumentation schemes which represent normative patterns of presumptive reasoning are more difficult to explain [34]. On this topic, we have some of the following general questions. What are the conditions on semantic well-formedness for the rules provided to the knowledge base? What makes a set of statements semantically cohere as premises of a claim? What makes a set of statements semantically imply a conclusion? While the mathematical or syntactical conceptions of a rule may be clear, and logicians or knowledge engineers may implicitly know how to write well-formed rules, the principles which underlie the linguistic semantic instantiations are not clear. If we allow non-logicians or knowledge engineers to build rules, what guidance can be provided to build meaningful rules? That is, how can we make explicit what logicians and knowledge engineers implicitly know?

These topics touch directly on complex issues relating to the interface of human reasoning, language, and formal reasoning ([15] and [31]); indeed, in addition to the issues that arise about controlled languages, it is problematic to presume that participants can, without training, contribute to a normative *rational* discussion in a way which supports automated reasoning; a similar point applies to the reconstruction of arguments from texts by participants. In other words, a degree of literacy in operating the system and interpreting the results is going to be required.

A related point bears on enthymemes, which are implicit premises that nonetheless bear on the inference ([7] and [35]). There are two

[9] ACE translates sentences such as *Every supermarket creates some garbage.* as generally done in First-order Logic, $\forall$x[supermarket(x) $\rightarrow$ $\exists$y[garbage(y) $\wedge$ create(x,y)]]. However, generalised quantifiers are the more desirable translation of quantifiers for natural language [2]. ACE also translates conditional statements as conditionals, though we have not included conditional statements among our rules in order to simplify. This does not impact on the key points in the paper.

sides to the issue – how enthymemes are used in reasoning and how they are incorporated into the knowledge base. The former presumes a solution to the latter since enthymemes are *implicit* in reasoning, which implies that they are *explicitly* represented in the knowledge base in some way. We do not address the former since in developing a knowledge acquisition device, we are concerned with soliciting implicit knowledge, making it explicit, and formally incorporating it into the knowledge base.

While all three of these issues are substantial in and of themselves, they are partially addressed in our approach to collaborative, incremental, distributive knowledge base construction. For example, suppose there are two sentences in the knowledge base which are mutually incompatible (e.g. our *Bill exercises daily* and *Bill is a couch potato*) but were not overtly recognised or marked as such by users; this could arise simply because of what the users attend to or the size of the knowledge base. Some subsequent users could (on separate occasions and without being aware of the input of other participants) introduce the relevant implications to yield the representation of contradiction. Similarly, rules which are constructed by some parties could subsequently be rendered inapplicable by asserting that a premise does not hold. Finally, as implicit knowledge is solicited, it is explicitly introduced into the knowledge base which can then support further explicit reasoning. In this industrialised vision of knowledge engineering, individual participants may be logically imperfect and partially ignorant, yet contribute to the construction of globally perfect knowledge bases which arise after the calculation of argument extensions. In this regard, the system does not so much rule out erroneous statements or rules as to allow them to be modified by the input of other participants and, more importantly, reasoned with to determine consequences which would highlight the problems.

Finally, we should touch on one representational issue. At different stages of the pipeline, different forms of representation are useful to support distinct reasoning processes. There is a balance to be struck at every step between abstraction and expressiveness. For instance, while we have used lower case letters and numbers, we could have used the sentence strings themselves as propositional variables. While we have left resolution of these issues to future work, it is not particularly problematic so long as we are clear about and consistent about the conversions at each state.

3 INSTANTIATING ARGUMENTS AND RELATIONSHIPS FROM A KNOWLEDGE BASE

In this section, we discuss instantiating arguments and their relationships from a *literals and rules knowledge base* (LRKB) as given above; the result is an *argument knowledge base* (AKB). The AKB is then input to an abstract argumentation framework to calculate extensions as discussed in Section 4; the result is an *argument extensions knowledge base* (AEKB), from which we can extract consistent sets of propositions which represent policy positions.

We work with a *logic-based* approach of [5] and [6], which represents arguments in terms of classical logic (for related work see ([28], [19], [21], [1], [5], and [27]). While we could discuss other approaches to instantiating arguments and relationships which use defeasible rules (e.g. [27] or [20]), we keep to a logic-based approach for several reasons: it is founded in a well-known and widely used logic (classical propositional logic), it has an extension to First-order Logic, it is broadly compatible with the logical translations provided by *ACE* (which has no defeasible rules), an implementation is available, and issues about generating and structuring arguments in relations are well-developed (e.g. minimal arguments, redundancy, and argument tree pruning among others). However, as we are primarily interested to explore an implemented example, we do not examine these issues further.

[5] and [6] provide numerous examples of knowledge bases and instantiated arguments primarily in Propositional Logic. The novelty of our example is twofold: our example is an abstraction from input sentences which have been automatically translated to First-order Logic; we have related the arguments and relations to an argumentation framework, where, as we see, additional issues arise.

Instantiating arguments and their relationships is necessary for calculating extensions of arguments, which are arguments which are mutually consistent. Abstract argumentation frameworks (AFs), which we briefly discuss in Section 4, define how to determine such extensions. Though [6] discuss AFs, in this paper, we take the primary focus of logic-based approach to be on instantiating arguments and their relationships. We review key components of the logic-based approach.

In a logic-based approach, statements are expressed as atoms (lower case roman letters), while formulae (greek letters) are constructed using the logical connectives of conjunction, disjunction, negation, and implication. The classical consequence relation is denoted by $\vdash$. Given a knowledge base Δ comprised of formulae and a formula α, $\Delta \vdash \alpha$ denotes that Δ entails α. Δ can be inconsistent and comprised of a range of declarative statements. We assume a set of formulae Δ from which arguments are constructed. Where $\perp$ denotes inconsistency, $\Delta \vdash \perp$ denotes that Δ is inconsistent. An argument is an ordered pair $< \phi, \alpha >$, where $\phi \subseteq \Delta$, ϕ is a minimal set of formulae such that $\phi \vdash \alpha$, and $\phi \nvdash \perp$. ϕ is said to support the claim α. For example, where p and q are atoms, and where the KB is comprised of p and $p \rightarrow q$, then $< \{p, p \rightarrow q\}, q >$ is an argument, where $p, p \rightarrow q$ is the support for the claim q.

The knowledge base Δ may be inconsistent, which here arises where Δ contains contradictory propositions (and not necessarily just constraints); this bears on issues discussed in Section 2.3. With contradictory propositions, we can construct arguments in relations, where the propositional claim of an argument is contradictory to the propositional claim of another argument or is contradictory to some proposition in the support of another argument. These are *attack* relations between arguments $< \Psi, \beta >$ and $< \Phi, \alpha >$ such as *undercutter* and *rebuttal*; attacking arguments are referred to as *counter-arguments*. $< \Psi, \beta >$ is an undercutter for $< \Phi, \alpha >$ where β is $\neg(\phi_1 \wedge \ldots \wedge \phi_n)$ and $\{\phi_1 \ldots \phi_n\} \subseteq \Phi$; in essence, the claim of one argument is the negation of a set of formulae in the support of another argument.[10] $< \Psi, \beta >$ is a rebuttal for $< \Phi, \alpha >$ if and only if $\beta \leftrightarrow \neg\alpha$ is a tautology; the claims of the arguments are inconsistent. For example, supposing the following LRKB (from [6]): $p, p \rightarrow \neg q$, $r, r \rightarrow \neg p$, $\neg p \rightarrow q$. From this LRKB, we can construct an argument to support the claim $\neg q$: $< \{p, p \rightarrow \neg q\}, \neg q >$. With respect to this argument, we have an undercutter $< \{r, r \rightarrow \neg p\}, \neg p >$ and a rebuttal $< \{r, r \rightarrow \neg p, \neg p \rightarrow q\}, \neg p >$.

Given a large and complex LRKB, arguments will have structural relationships such as subsumption of supports, where one support is a subset of another support, and implication between claims, where one claim entails another. Moreover, there may be more than one argument which undercuts or rebuts another argument. [5] and [6] define and discuss a range of these relationships among arguments; however, additional definitions are not directly relevant to our key

[10] There is an additional notion of *canonical undercut*, where the atoms are ordered; it is useful for efficiency. For the presentation here, we presume it.

points in this paper. For our purposes, given an LRKB, we can generate not only the arguments, but also the counterarguments, the counterarguments to these arguments (counter-counterarguments), and so on recursively; such a structure is an *argument tree*, a graph where arguments are nodes and attack relations are (undifferentiated) arcs. From a given LRKB, [5] generate all possible arguments and counterarguments. To illustrate this with an example, we discuss *JArgue*, which implements the logic-based approach.

3.1 JArgue

JArgue is an implementation in Java of logic-based argumentation [5]; given an LRKB, JArgue generates arguments, attacks between arguments, and an argument tree.[11] One provides JArgue with an LRKB as a text file of propositional clauses (*conjunctive normal form*); implicational rules such as $p \rightarrow q$ are rewritten as $!p|q$, where ! is negation and | is disjunction. Where the antecedent of a rule is a conjunction of literals as in $[r \wedge s] \rightarrow q$, we rewrite the negation of the antecedent as a disjunct of negated literals – $!r|!s|q$. Starting JArgue and inputting an LRKB, one can request the generation of arguments where a particular literal is the claim; given the arguments, one can then select among the arguments and request the generation of the argument tree. We discuss JArgue and illustrate it with our example.

Our example in Section 2.1 has what we are treating as atoms p1-p19 and rules r1-r8. Several preliminary comments are needed before presenting the input LRKB. First, since incompatible literals is a significant issue, we did not presume to represent statements which were incompatible as logical contradictions, e.g. p1 and p2, p11 and p18, and p15 and p17, even though ACE translates these pairs as contradictory. In other words, we presumed the problematic examples of contradiction rather than the unproblematic.

p1: Every household should pay some tax for the households garbage.
p2: No household should pay some tax for the households garbage.
p11: Tom is not an objective expert about recycling.
p18: Tom is an objective expert about recycling.
p15: Every supermarket should pay a tax for the garbage that the supermarket creates.
p17: No supermarket should pay a tax for the garbage that the supermarket creates.

However, to determine attack relations, the logic-based approach requires the explicit expression of contradiction as in classical logic. We therefore make the following substitutions: p2 $\equiv \neg$p1, p11 $\equiv \neg$p18, and p17 $\equiv \neg$p15. This is an issue relevant to the illustration, not to the functionality of either the logic-based approach or JArgue since expressions of contradiction can be added to the knowledge base as it is built. However, the choice here is to represent such incompatibilities as logical contradiction rather than as constraints of the form $\neg$[p1 $\wedge$ p2] (or in conjunctive normal form for JArgue !p1|!p2). We explore this choice in future research.

Given these equivalences, the related rules are adjusted by substituting in equivalent expressions:

r1: [p4 $\wedge$ p3] $\rightarrow$ p1
r2: [p9 $\wedge$ p18 $\wedge$ p19] $\rightarrow$ p4
r3: p10 $\rightarrow$ p3
r4: [p12 $\wedge$ p13] $\rightarrow \neg$p18
r5: [p5 $\wedge$ p15] $\rightarrow \neg$p1
r6: [p6 $\wedge$ p7 $\wedge$ p8] $\rightarrow$ p5
r7: p14 $\rightarrow$ p15
r8: p16 $\rightarrow \neg$p15

Our LRKB for JArgue will be comprised of literals and (conjunctive normalised) rules. However, there is a somewhat tangential issue of whether we need include *all* literals since some serve only as *intermediaries* between rules (see the discussion of *intermediate concepts* in [36]); that is, if we have a language comprised of p, q, r, $p \rightarrow q$, $q \rightarrow r$, there is no reason to include in the KB q (or r) itself since it is enough to include all the rules and only those literals which are terminal leaves in a graph of *all the rules* as in Figure 1; in other words, the KB need only represent what is *asserted* along with the rules, not what is simply expressed in the language. In a dynamically constructed KB, what is asserted and what is inferred may change over time. We presume this simplification and leave further discussion to future research. Thus, we have rules r1-r8 as revised above and only p6, p7, p8, p9, p10, p12, p13, p14, p16, p18, and p19. Other literals are inferred.[12]

With these points in mind, our input LRKB for JArgue is as follows:

p6, p7, p8, p9, p10, p12, p13, p14, p16, p18, p19, !p10|p3, !p9|!p18|!p19|p4, !p12|!p13|!p18, !p3|!p4|p1, !p5|!p15|!p1, !p6|!p7|!p8|p5, !p14|p15, !p16|!p15

To run JArgue, we add this LRKB, then query the KB to generate an argument. Querying the KB can only be done for one literal at a time. For our example, if we query for p1, we get the following argument, which we label a1:[13]

a1: $< \{p10, p9, p18, p19, !p10|p3, !p9|!p18|!p19|p4, !p3|!p4|p1\}, p1 >$

It happens to be the case that there is only one argument a1 for p1 generated from our KB, but with another KB, there could have been more.

JArgue then requests the user to select from an argument with respect to which it should generate an argument tree. Selecting a1, JArgue generates an argument tree with two counterarguments a2 and a3 along with one counterargument a4 to a3; for the sake of discussion, we discuss each node and their relationship rather than present the graph. The first counterargument a2 is an undercutter of a1 since the claim of a2, !p18, is the negation of a literal which appears in the support of a1:[14]

a2: $< \{p12, p13, !p12|!p13|!p18\}, !p18 >$

The second counterargument a3 is a rebuttal of a1 since the claim of a3, !p1, is the negation of the claim of a1:

a3: $< \{p6, p7, p8, p14, !p6|!p7|!p8|p5, !p14|p15$ $!p5|!p15|!p1\}, !p1 >$

[11] Requests for JArgue should be addressed to Tony Hunter a.hunter@cs.ucl.ac.uk.

[12] This may be a more substantive issue since what arguments and relations are generated and used for input to the abstract framework may depends on what is explicitly available in the knowledge base rather than the explicit and inferred statements. We discuss this further below, and it may need further examination.

[13] Labelling the arguments is not a functionality of Logic-based argumentation or JArgue. We do so to provide elements for the AEKB.

[14] Logic-based argumentation and JArgue represent the claim of a canonical undercutter with *; for the purposes of our illustration, we make it explicit.

Finally, we have a counterargument to a3, where one of the implied literals of a3, p15, is contradicted by the claim of a4, !p15:

a4: $< \{p16, !p16|!p15\}, !p15 >$

If we query the LRKB for !p1 and keep the argument labeling the same, we have a3. The argument tree for a3 gives us a1 as a counterargument; this shows that where we have contradictory *claims*, the arguments attack one another. Otherwise, a2 is a counterargument of a1, and a4 is a counterargument for a3 as before.

Querying just the argument trees for p1 and !p1, we can create an AKB, which is a knowledge base of generated arguments in attack relations (indicated with *att*), which is given in the argument tree. Such an AKB is suitable for input to an abstract argumentation framework to generate an AEKB, which allows us to reason abstractly over arguments. This is AKB1:

Arguments: a1, a2, a3, a4
Attack Relations: att(a1,a3), att(a3,a1), att(a2,a1), att(a4,a3)

3.2 Considerations

Where we choose to argue for and against literals at the root of an argument tree such as about p1 and !p1, the arguments and relations among arguments appear relatively straightforward. It is important to note that we have used JArgue to generate argument trees with respect to a query on a proposition. However, we are using it as well to creating AKBs for input to abstract argumentation frameworks, which is the reason to investigate the argument trees for a proposition *and* its negation; however, using JArgue to generate arguments and relations to support abstract argumentation frameworks is somewhat beyond the brief, as we see in a moment.

A relevant general question arises – how many arguments and attack relations among them can be generated from the underlying LRKB? Given that we have classical logic, *contrapositive* reasoning applies; that is, given ¬q and $p \rightarrow q$, we can infer ¬p. Thus, if we query p3, which is implied by p10, which is itself asserted, we can generate an argument and an argument tree.

a': $< \{p10, !p10|p3\}, p3 >$

In the argument tree, we have a rebuttal argument which is generated by contraposition – where !p1 and p4 hold, !p3 must hold:

a'': $< \{p6, p7, p8, p9, p14, p18, p19, !p6|!p7|!p8|p5, !p14|p15$
$!p9|!p18|!p19|p4, !p5|!p15|!p1, !p3|!p4|p1\}, !p3 >$

This argument itself has two counterarguments. And in particular, a' and a'' rebut one another. More generally, every literal in the language (appearing in a rule or as an assertion) has an argument and a counterargument (each of which may also have counterarguments). Given this, should all such arguments in their relations be in the AEKB for a given AKB?

For our purposes, we only include those arguments and relations relative to a debate under investigation, here p1 and !p1; these are the arguments and relations as generated by JArgue. Other arguments and counterarguments such as for p3 are not clearly relevant. Indeed, perhaps all the other arguments and relations arise via contraposition are problematic; indeed, it is questionable whether contraposition is a *natural* reasoning rule (see [15] and [31]).

The hedge, however, bears on how we use AKB as input to the abstract argumentation framework discussed in Section 4, where we find the AKB with four arguments and four attacks is not sufficient to get a correct result. This is so because while a2 attacks a1, and a4 attacks a3, nothing attacks a2 or a4; thus, these arguments attack and defeat a1 and a3. The extensions leave just a2 and a4, which is contrary to our desired goal, which is that a1 and a4 could hold together, while a3 and a2 could hold together. We need, then, to generate an attack on a2 where we query for the claim of a1, and to generate another attack on a4 when we query for the claim of a3.

This would seem to require that where we query for a proposition and its negation, we want to generate at least one additional argument to counterattack attackers, if we can. In other words, we want to generate attacks (either rebuttals or undercutters) to a2 and a4. For example, we could do so by finding a rebuttal which uses (in a sense to be explained) the claim of the queried argument. In other words, we want to find one rebuttal a5 where the claim of a5 is the negation of the claim of a2 and where the claim of a1, namely p1, is used is found among the support for the claim of a5; similarly, we want to find another rebuttal a6 where the claim of a6 is the negation of the claim of a4 and where the claim of a3, namely ¬p2, is used. Such arguments may be generated from the underlying LRKB or they may be generated by contraposition, which then provides some intuitive motivation to support such a rule in the context of argumentation. In other words, we want to find instantiated arguments to realise AKB2:

Arguments: a1, a2, a3, a4, a5, a6
Attack Relations: att(a1,a3), att(a3,a1), att(a2,a1), att(a4,a3), att(a5,a2), att(a2,a5), att(a6,a4), att(a4,a6)

The result is a graph as in:

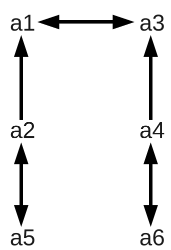

Figure 2. Arguments and Relationships

Note that arguments a1-a4 are clearly related to our earlier representation of the relationships among statements in Figure 1, but that a5 and a6 are not.

However, to find a5 and a6, we have some additional issues to consider, in particular the argument trees of each of these arguments. We consider the issues by example. a2 has as claim ¬p18, thus the rebuttal a5 must have p18 as claim. In fact, we have an argument by assertion for this claim: <{p18}, p18>. However, this has a counterargument (derived by contraposition) as well as a counter-counterarguments:

$< \{p6, p7, p8, p14, p10, !p6|!p7|!p8|p5, !p14|p15, !p10|p3,$
$!p5|!p15|!p1, p9, p19, !p3|!p4|p1, !p9|!p18|!p19|p4\}, !p18 >$

Must we include, then, the whole argument tree for a5? However, we observe that p1 and !p1 are used in this argument. If p1 is in fact asserted (given that is the argument we query), then this counterargument to a5 (and the counter-counterargument) can no longer be generated, leaving a5 <{p18}, p18>. By the same token, we need an argument to defend p15 from attack by argument a4; this is a6 $< \{p14, !14|15\}, p15 >$

While these are remarks about what are required to have a AKB suitable for reasoning with AFs, we leave to future research how this is theoretically developed and implemented. In any case, adopting these additional arguments gives more satisfactory results in terms of AFs. The important point is that we must carefully consider what results we might want with respect to AFs in order to create the appropriate arguments and relations, then build into our argument generator the mechanisms to generate the arguments and relations, otherwise the desired goal from the AF will not be achieved.

4 FROM INSTANTIATED ARGUMENTS TO AN AF

Having generated an AKB of arguments and their relations, we can input it to an abstract argumentation framework AF to calculate extensions. In this section, we review basic components of AFs, outline results with respect to an implementation, and then discuss the results.

4.1 AFs

For our purposes, we consider the AF of [14], where there is one set of undifferentiated objects, *arguments*, which are nodes in a graph as well as one undifferentiated relationship between the nodes, the *attack* relation, which can be represented as a graph in which attacks are arcs between nodes representing the arguments (for related work, see [8] [3], [10], among others).

Definition An argumentation framework AF is a pair $< \mathcal{X}, \mathcal{R} >$, where $\mathcal{X}$ is a set of *objects*, $\{a_1, a_2, \ldots, a_n\}$ and $\mathcal{R}$ is an *attack* relation between objects. For $\langle a_i, a_j \rangle \in \mathcal{R}$ we say the the object a_1 attacks object a_2. We assume that no object attacks itself.

Some of the relevant auxiliary definitions are as follows, where S is a subset of $\mathcal{X}$. There are a range of *semantics*, which are definitions of extensions (sets) of arguments; we give only admissible and preferred semantics:

Definition We say that $p \in \mathcal{X}$ is *acceptable with respect to S* if for every $q \in \mathcal{X}$ that attacks p there is some $r \in S$ that attacks q. A subset, S, is *conflict-free* if no argument in S is attacked by any other argument in S. A conflict-free set S is *admissible* if every $p \in S$ is acceptable to S. A *preferred extension* is a maximal (w.r.t. $\subseteq$) admissible set.

In the next section, we use the ASPARTIX system to generate argument extensions.

4.2 CALCULATING EXTENSIONS WITH *ASPARTIX*

The ASPARTIX program computes extensions of argumentation frameworks [17].[15] It is, then, very useful for our purposes, where we have created arguments in relations and then wish to compute extensions. ASPARTIX is implemented in the answer set programming language *Disjunctive Datalog System* (DLV).[16] Using ASPARTIX, one can compute any of the standard extensions for the classical AF (following Dung) including admissible and preferred extensions. Extensions for preference-based, value-based, and bipolar argumentation frameworks can also be generated.

To run ASPARTIX, we input to the program a text file for the AKB, containing the arguments and relations, as well as the desired sort of extension. We used our previously defined AKB, which has the arguments and attack relations:

```
arg(a1).                att(a3,a1).
arg(a2).                att(a2,a1).
arg(a3).                att(a4,a3).
arg(a4).                att(a5,a2).
arg(a5).                att(a2,a5).
arg(a6).                att(a6,a4).
att(a1,a3).             att(a4,a6).
```

ASPARTIX generates the following preferred extensions, where *in* indicates that the argument is in the extension:

```
{in(a1), in(a5), in(a6)}
{in(a2), in(a3), in(a6)}
{in(a3), in(a5), in(a6)}
{in(a1), in(a4), in(a5)}
{in(a2), in(a4)}
```

Such extensions which are generated by the AF we refer to as the *Argument Extensions Knowledge Base* (AEKB).

This seems to be an attractive result. As we are only interested in extensions with *p1* and $\neg$*p1*, we will not look at {in(a2), in(a4)}. As a remark, preferred extensions seem to be the argumentation correlate to the notion *normative positions* of deontic logic [29], which are maximal sets of consistent deontic expressions. In light of this, we will call them *policy positions*.

5 Extracting Policy Positions

Our sets of arguments are an abstract representation of the contents of the arguments themselves, which are comprised of literals and rules. Thus, given the sets of arguments, we can extract the literals and rules that are consistent and which represent the correlated linguistic expressions of the policy positions. While we can reconstruct the content of arguments a1, a2, a3, and a4, the additional arguments for a5 and a6 are less clear. We will suppose that these are, as we suggested, generated as rebuttals.

Recall that our arguments are:

a1: $< \{p10, p9, p18, p19, !p10|p3, !p9|!p18|!p19|p4, !p3|!p4|p1\}, p1 >$
a2: $< \{p12, p13, !p12|!p13|!p18\}, !p18 >$
a3: $< \{p6, p7, p8, p14, !p6|!p7|!p8|p5, !p14|p15, !p5|!p15|!p1\}, !p1 >$
a4: $< \{p16, !p16|!p15\}, !p15 >$
a5: $< \{p18\}, p18 >$
a6: $< \{p14, !14|15\}, p15 >$

Let us informally define a policy position (PP) as supports and claims of all the arguments in the extension (we could do without the claim, but in a sense, the objective is making explicit what is

[15] We thank the ASPARTIX team, and particularly Sarah Gaggl, for making the ASPARTIX system available to us. See `http://www.dbai.tuwien.ac.at/staff/gaggl/systempage/index.html`.

[16] See `http://www.dbai.tuwien.ac.at/proj/dlv/`.

otherwise implicit and inferred). Formally, we have two functions: F1 is a function on the extension such that it outputs the union of the result of the application of F2 on every argument of the extension; F2 is a function on arguments which outputs a set comprised of the union of the support with the claim. We suppose these functions. Furthermore, since we have sets, any redundancy is removed.

PP1: $\{p1, p4, p9, p10, p14, p15, p18, p19, !p10|p3, !p9|!p18|!p19|p4, !p3|!p4|p1, !p14|p15\}$
PP2: $\{!p1, p5, p6, p7, p8, p12, p13, p14, p15, !p18, !p12|!p13|!p18, !p6|!p7|!p8|p5, !p14|p15, !p5|!p15|!p1\}$
PP3: $\{!p1, p5, p6, p7, p8, p14, p15, p18, !p6|!p7|!p8|p5, !p14|p15, !p5|!p15|!p1\}$
PP4: $\{p1, p3, p4, p9, p10, !p15, p16, p18, p19, !p10|p3, !p9|!p18|!p19|p4, !p3|!p4|p1, !p16|!p15\}$

The propositions and rules here correlate directly with the propositions and rules introduced in Section 2.1, so we can substitute them here. We will put in the sentences and indicate the rules for PP2:

PP2 Sentences and Rules
!p1: No household should pay some tax for the household's garbage.
p5: If a household pays a tax for the households garbage then the tax is unfair to the household.
p6: Every household should pay an equal portion of the sum of the tax for the households garbage.
p7: No household which receives a benefit which is paid by a council recycles the households garbage.
p8: Every household which does not receive a benefit which is paid by a council supports a household which receives a benefit which is paid by a council.
p12: Tom owns a company that recycles some garbage.
p13: Every person who owns a company that recycles some garbage earns some money from the garbage which is recycled.
p14: Every supermarket creates some garbage.
p15: Every supermarket should pay a tax for the garbage that the supermarket creates.
!p18: Tom is not an objective expert about recycling.
r4: [p12 ∧ p13] → !p18
r5: [p5 ∧ p15] → !p1
r6: [p6 ∧ p7 ∧ p8] → p5
r7: p14 → p15

This seems an accurate representation of this policy position from the onset. While PP2 and PP4 seem acceptable representations of policies.

5.1 CONSIDERATIONS

The policy positions of PP1 and PP3 seem to have some additional superfluous information. For instance, PP1 contains the propositions p14 and p15, which are propositions in the support of !p1, which itself is the rebuttal of the main point of PP1. Since other portions of the support for !p1 do not hold, !p1 does not hold. Yet, it would seem, in an intuitive argument, odd to have as part of one's position statements in support of the opposite point, even if these statements are not inconsistent with one's position. We see, then, that AFs *overgenerate* policy positions. In addition, the output sentences of the policy positions are without discourse markers, so the relations among them are unclear. We leave for future research just how to refine the results to generate all and only the policy positions that correlate with an intuitive notion as well as how to use natural language generation technology to provide well-formed, easy-to-understand output paragraphs of the policy positions.

6 DISCUSSION

In this paper, we have shown how statements expressed in natural language can be represented as a formal knowledge base of literals and rules (LRKB), that the LRKB can be fed into a framework for generating arguments and relations to create an knowledge base of arguments (AKB), and then in turn, the AKB can be provided to an AF to generate a set of extensions which is the knowledge base of extensions of arguments (AEKB). The extensions we related to policy positions, which are consistent sets of statements and rules; we represented the policy positions in terms of the underlying source statements. At each point of the transformation, we discussed relevant issues that arose.

The main novelty of our approach is that formal, implemented approaches to the syntax and semantics of natural language have been used to develop the LRKB, that we have used implemented systems for generating AKB and AEKB, and that we have identified some important novel issues and problems for future research in the development of the argument pipeline. In these respects, our approach is distinct from similar work such as [5], [6], [26], and [38], where disputes expressed in natural language are translated manually into AKB and AEKB. While manual analysis can work, it does not address the problem of the knowledge acquisition bottleneck, nor does it make explicit, formal, or implemented the knowledge engineer's knowledge in making the analysis.

In addition to observations made in the *considerations* sections, the work reported here does not discuss a range of other interesting and relevant issues which we seek to address in the future. For instance, we have left aside any issues relating to knowledge base or context dynamics. Rather, as with classical logics, we have presumed static knowledge bases and fixed contexts. This is entirely a strategic move in order to pin down interesting issues about the static domain per se which are also relevant to dynamic system and context. The approach also emphasises dialectical argumentation such as found in legal disputes, where there is one specific statement being argued for and against and where we generate policy positions. There are, as we know, a range of other dialogical modes and goals, e.g. deliberation, information gathering, and consensus building (see, for example, [33]). Nonetheless, it all of these dialogues, it is fundamental to identify and work with *difference* and *inconsistency*, whether to resolve it, eliminate it, or to adapt to it. Different dialogical modes and goals may be thought of as different strategies to work with difference and inconsistency. Finally, the approach we have taken here makes use of *rhetorical relationships*, e.g. premise, claim, and contradiction, between statements that suit a literals and rules knowledge base that is compatible with Propositional or First-order Logic. However, there are other rhetorical relationships that are relevant to dialogue and argumentation which ought to be incorporated into future analysis and implementations (see [24], [22], and [30]).

ACKNOWLEDGEMENTS

The first two authors thank the European Commission for supporting this research in the IMPACT Project (Integrated Method for Policy making using Argument modeling and Computer assisted Text analysis) FP7 Grant Agreement No. 247228.

REFERENCES

[1] Leila Amgoud, Martin Caminada, Claudette Cayrol, Marie-Christine Lagasquie, and Henry Prakken, 'Towards a consensual formal model: inference part', Technical report, ASPIC project, (2004). Deliverable D2.2: Draft Formal Semantics for Inference and Decision-Making.

[2] Jon Barwise and Robin Cooper, 'Generalized quantifiers and natural language', *Linguistics and Philosophy*, **4**, 159–219, (1981).

[3] Trevor J. M. Bench-Capon, 'Persuasion in practical argument using value-based argumentation frameworks.', *Journal of Logic and Computation*, **13**(3), 429–448, (2003).

[4] Philippe Besnard and Anthony Hunter, 'Practical first-order argumentation', in *AAAI*, eds., Manuela M. Veloso and Subbarao Kambhampati, pp. 590–595. AAAI Press / The MIT Press, (2005).

[5] Philippe Besnard and Anthony Hunter, *Elements of Argumentation*, MIT Press, 2008.

[6] Phillip Besnard and Anthony Hunter, 'Argumentation based on classical logic', in *Argumentation in Artificial Intelligence*, eds., Iyad Rahwan and Guillermo Simari, 133–152, Springer, (2009).

[7] Elizabeth Black and Anthony Hunter, 'Using enthymemes in an inquiry dialogue system', in *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'08)*, pp. 437–444. ACM Press, (2008).

[8] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni, 'An abstract, argumentation-theoretic approach to default reasoning', *Artificial Intelligence*, **93**, 63–101, (1997).

[9] Emma Bryne and Anthony Hunter, 'Man bites dog: Looking for interesting inconsistencies in structured news reports', *Data and Knowledge Engineering*, **48**(3), 265–295, (2004).

[10] Martin Caminada and Leila Amgoud, 'On the evaluation of argumentation formalisms', *Artificial Intelligence*, **171**(5-6), 286–310, (2007).

[11] Alan Cruse, *Lexical Semantics*, Cambridge University Press, 1986.

[12] Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning, 'Finding contradictions in text', in *Proceedings of ACL-08: HLT*, pp. 1039–1047, Columbus, Ohio, (June 2008). Association for Computational Linguistics.

[13] Valeria de Paiva, Daniel G. Bobrow, Cleo Condoravdi, Dick Crouch, Tracy H. King, Lauri Karttunen, Rowan Nairn, and Annie Zaenen, 'Textual inference logic: Take two', in *C&O:RR*, (2007).

[14] Phan Minh Dung, 'On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games', *Artificial Intelligence*, **77**(2), 321–358, (1995).

[15] Dorothy Edgington, 'Conditionals', in *Stanford Encyclopedia of Philosophy*, ed., Edward Zalta, Stanford University, (2006). http://plato.stanford.edu/entries/conditionals/.

[16] Vasiliki Efstathiou and Anthony Hunter, 'An algorithm for generating arguments in classical predicate logic', in *ECSQARU*, pp. 119–130, (2009).

[17] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran, 'Answer-set programming encodings for argumentation frameworks', *Argument and Computation*, **to appear**, to appear, (2008).

[18] Diana E. Forsythe and Bruce G. Buchanan, 'Knowledge acquisition for expert systems: some pitfalls and suggestions', in *Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems*, 117–124, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, (1993).

[19] Alejandro Javier García and Guillermo Ricardo Simari, 'Defeasible logic programming: An argumentative approach', *Theory and Practice of Logic Programming*, **4**(1), 95–137, (2004).

[20] Thomas Gordon, Henry Prakken, and Douglas Walton, 'The carneades model of argument and burden of proof', *Artificial Intelligence*, **171**, 875–896, (2007).

[21] Guido Governatori, Michael J. Maher, Grigoris Antoniu, and David Billington, 'Argumentation semantics for defeasible logic', *Journal of Logic and Computation*, **14**(5), 675–702, (2004).

[22] Floriana Grasso, 'Towards a framework for rhetorical argumentation', in *Proceedings of the 6th Workshop on the Semantics and Pragmatics of Dialogue (EDILOG-2002)*, eds., Johan Bos, M. E. Foster, and C. Matheson, pp. 53–60, Edinburgh, UK, (2002).

[23] Ann Macintosh, 'Moving toward "intelligent" policy development', *IEEE Intelligent Systems*, **24**(5), 79–82, (2009).

[24] William C. Mann and Sandra A. Thompson, 'Rhetorical structure theory: A theory of text organization', Technical Report ISI/RS-87-190, University of Southern California, Information Sciences Institute (ISI), (June 1987).

[25] Rudy Prabowo and Michael Thelwall, 'Sentiment analysis: A combined approach', *Journal of Informetrics*, **3**(1), 143–157, (2009).

[26] Henry Prakken, 'Formalising ordinary legal disputes: a case study', *Artificial Intelligence and Law*, **16**, 333–359, (2008).

[27] Henry Prakken, 'An abstract framework for argumentation with structure arguments', *Argument and Computation*, **?**, ?–?, (2010). To appear.

[28] Henry Prakken and Giovanni Sartor, 'Argument-based extended logic programming with defeasible priorities', *Journal of Applied Non-Classical Logics*, **7**(1), (1997).

[29] Marek Sergot, 'Normative positions', in *Norms, Logics, and Information Systems*, eds., McNamara and Prakken, 289–308, IOS Press, (1998).

[30] Caroline Sporleder and Alex Lascarides, 'Using automatically labelled examples to classify rhetorical relations: An assessment', *Natural Language Engineering*, **14**(3), 369–416, (2006).

[31] Keith Stenning and Michiel van Lambalgen, 'A little logic goes a long way: basing experiment on semantic theory in the cognitive science of conditional reasoning', *Cognitive Science*, **28**(4), 481–529, (2004).

[32] Ellen M. Voorhees, 'Contradictions and justifications: Extensions to the textual entailment task', in *Proceedings of ACL-08: HLT*, pp. 63–71, Columbus, Ohio, (June 2008). Association for Computational Linguistics.

[33] Doug Walton, Katie Atkinson, Trevor Bench-Capon, Adam Wyner, and Doug Cartwright, 'Argumentation in the framework of the deliberation dialogue', in *Arguing Global Governance: Agency, Lifeworlds and Shared Reasoning*, eds., Corneliu Bjola and Markus Kornprobst, Routledge, (2010). Forthcoming.

[34] Douglas Walton, *Argumentation Schemes for Presumptive Reasoning*, Erlbaum, Mahwah, N.J., 1996.

[35] Douglas Walton, 'The three bases for the enthymeme: A dialogical theory', *Journal of Applied Logic*, **6**(3), 361–379, (2008).

[36] Adam Wyner, 'An ontology in OWL for legal case-based reasoning', *Artificial Intelligence and Law*, **16**(4), 361–387, (December 2008).

[37] Adam Wyner, Krasmir Angelov, Guntis Barzdins, Danica Damljanovic, Brian Davis, Norbert Fuch, Stefan Hoefler, Ken Jones, Kaarel Kaljurand, Tobias Kuhn, Martin Luts, Jonathan Pool, Mike Rosner, Rolf Schwitter, and John Sowa, 'On controlled natural languages: Properties and prospects', in *Workshop on Controlled Natural Languages, CNL 2009*, ed., Norbert E. Fuchs, number 5972 in LNCS/LNAI. Springer, (2010). To appear.

[38] Adam Wyner, Trevor Bench-Capon, and Katie Atkinson, 'Arguments, values and baseballs: Representation of popov v. hayashi', in *Legal Knowledge and Information Systems. JURIX 2007*, eds., Arno R. Lodder and Laurens Mommers, pp. 151–160, Amsterdam, (2007). IOS Press.

[39] Adam Wyner, Raquel Mochales-Palau, Marie-Francine Moens, and David Milward, 'Approaches to text mining arguments from legal cases', in *Semantic Processing of Legal Texts*, eds., Simonetta Montemagni, Daniela Tiscornia, and Enrico Francesconi, Lecture Notes in Computer Science, Springer, (2009). To Appear.

[40] Adam Wyner, Tom van Engers, and Kiavash Bahreini, 'From policy-making statements to first-order logic', in *Proceedings of International Conference on Electronic Government and the Information Systems Perspective (EGOVIS 2010)*, Lecture Notes in Computer Science. Springer Verlag, (2010). To appear.