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

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

[1] Department of Computing Science, University of Aberdeen, Aberdeen, United Kingdom
adam@wyner.info
[2] Leibniz Centre for Law, University of Amsterdam, Amsterdam, The Netherlands
vanEngers@uva.nl,
[3] Department of Computer Science, University College London, London, United Kingdom
a.hunter@cs.ucl.ac.uk

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 reports progress towards reasoning with web-based, distributed, collaborative, incomplete, and inconsistent knowledge bases expressed in natural language.

1 Introduction

In many domains of public discourse, participants have the opportunity to present their views.[4] In public policy-making discussions, members of the public and politicians

[4] Corresponding author: Adam Wyner adam@wyner.info. A version of this paper was previously presented at The ECAI 2010 workshop on Computational Models of Natural Argument, 16 August 2010, Lisbon, Portugal.

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 [2]. 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 for policy-making 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 [3]. 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. Along the way, knowledge engineers might interpret sentences relative to context as well as fill in the semantic representation with pragmatic background knowledge. 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. However, we decompose the overall problem in order to focus on the specific aspects - processing of statements with automated techniques in natural language processing, instantiated arguments, and reasoning with the inconsistent 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 (a literals and rules knowledge base – LRKB) using *Attempto Controlled English* (ACE)[5]; from this knowledge base, we generate instantiated arguments and their relationships (an argument knowledge base – AKB) using a tool grounded in the logic-based formalism of [4]; we input

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

this knowledge base into an implemented argumentation framework which calculates extensions of arguments (an argument extensions knowledge base AEKB) using [5] that is grounded in [6]. At the end, we extract consistent sets of expressions (policy positions) that represent a cohesive and coherent "view" on the policy proposal.[6] As indicated above, there are many issues bearing on knowledge engineering, and the scope of this paper addresses only some of the key issues.

While each of these components is developed independently, little consideration has been focused on *integration* and *through flow* between them. The important novelty of this paper is an identification and exploration of these bridge issues, which must be addressed to construct an on-line *argument processing pipeline* that starts with expressions in natural language that feed data to an implemented logic-based formalism that in turn feeds argumentation frameworks, more formally. The objective of the pipeline is to 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 without training in building well-formed rules, ill-formed arguments could be entered. What prompts can be introduced to make KB construction systematic and meaningful? What is the appropriate semantic formalism and grainedness for natural language (Propositional, First-order, Modal)? 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 simplifications that help 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 the paper, we discuss each of the processing modules in turn, including considerations and related work. 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 ACE tool. 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 forming 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.

[6] The term policy position is intended to be related to the *normative positions* of deontic logic [7].

2 Example, Translation Methodology, and Considerations

In this section, we discuss our source material and working example, the methodology of using the ACE tool 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

[8] 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.[7] For the particular study, [8] considered comments in response to the question *Should people be paid to recycle?*, where 697 comments were published (and 35 were rejected).[8]

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, no current computational linguistic or text-mining systems accurately parse the large majority of sentences, yield the acceptable correlated semantic interpretation, and can be used to support the acquisition of the knowledge for argument representation. Text-mining can extract some useful information (e.g. sentiment analysis [9], and some aspects of rules [10,11], and contradictions [12]), 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. Large corpora can be queried for extraction of argument elements, thought this is not parsing and semantic representation [13,14,15]. There are parsers with semantic interpretation that work on large scale [16], but require detailed examination and refinement to produce appropriate output [17]. Other parsers with semantic interpretation work on a fixed range of corpus-specific constructions [18,19]. There is work on textual entailment [20], some aimed at argumentation at a coarse-grained level [21,22]; however, this is insufficient to represent the internal structure of arguments and reconstruct relevant relationships. Uncertainties and inconsistencies

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

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

(other than explicit contradictions) are difficult to identify in current approaches. It remains a very significant, manual task to take a list of comments from a blog and transform them into a knowledge base that is suitable for a KR task. There has been intensive recent interest in and progress on *argument mining*, which is automatic extraction of structured arguments from unstructured textual documents (see [1] and citations therein). It remains highly challenging to extract the components of structured arguments and contrasting statements from source text.

Given the current state of the art in argument mining, an alternative approach to input of sentences and rules is to use a controlled language, e.g. ACE used here (also see [23,24]. This allows us to input homogenised, structured information with respect to the lexicon and syntactic forms, allowing inferences and contrast identification. We can take controlled expressions as an intermediate form between source unstructured text and what is needed for formal argument evaluation; expressions mined from unstructured text might be mapped to the intermediate form so as to facilitate argument evaulation. Moreover, the focus of this paper is on the flow through issues rather than NLP or sentence input *per se*.

[8] 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. By addressing relatively artificial examples, we can highlight issues, then extend the approach incrementally to solve additional and more natural problems and examples.

With these programmatic points in mind, we adopt and adapt the approach and sentences in [8], where 16 sentences are automatically translated into First-order logic (with some modal operators), 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 because the theory of argument formation for expressions in First-order logic is more complex than we need here and because the implemented system that we use for generating arguments and relations only works for Propositional Logic. 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.[9]

Example Sentences

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

p1: Every household should pay some tax for the household's garbage.
p2: No household should pay some tax for the household's garbage.
p3: Every household which pays some tax for the household's garbage increases an amount of the household's garbage which the household recycles.
p4: If a household increases an amount of the household's garbage which the household recycles then the household benefits the household's society.
p5: If a household pays a tax for the household's 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 household's garbage.
p7: No household which receives a benefit which is paid by a council recycles the household's 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 household's 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 household's 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 household's garbage reduces a need of a new dump which is for the garbage, then every household which recycles the household's garbage reduces a need of a new dump which is for the garbage.

As in [8], where a rule has more than one premise, we take it that the premises conjunctively hold for the claim to hold. The rules are *definite clauses* - Horn clauses with exactly one positive literal. 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

In addition to rules, we have what we express here as several constraints (or meaning postulates) on models which express contradiction between propositions. For example, p18 *Tom is an objective expert about recycling* and p11 *Tom is not an objective expert about recycling* are intuitively contradictory, so we write $\neg$p11 $\vee$ $\neg$p18. We discuss below further issues relating to contradiction

Constraints
c1: ¬p1 ∨ ¬p2
c2: ¬p11 ∨ ¬p18
c3: ¬p15 ∨ ¬p17

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

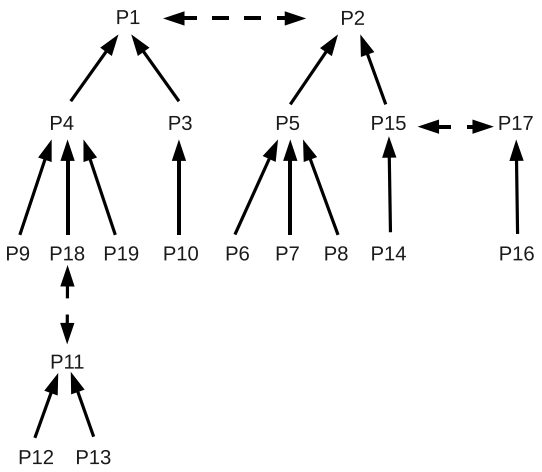

Fig. 1. Relationships between Statements – Premises, Claims, Contradictions

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 (ACE)

[8] provide a detailed discussion of the application of 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 (with some modal expressions) that can be used in future implemented argument instantiation systems [25,26].

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 [24]). 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 [8], the syntactic form must be *crafted* to generate the intended semantic interpretation (see further discussion below). Our 19 sentences are grammatically well-formed in ACE and have the intended interpretations.[10]

2.3 Considerations

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

As mentioned above, input to ACE must be crafted, which highlights the problem of engaging members of the public as knowledge engineers and using a natural language since this gives rise to a range of expectations which must be addressed. The user must accommodate to the restrictions of the language, which limits the spectrum of users of the system; however, research indicates the language is not unduly difficult to learn or use [28], university students (among others) could learn to use it. On the other hand, the system allows users to communicate in a form that they are most familiar with, and in this aspect the approach improves over other policy-making tools. As a controlled language, we work with a (large) fragment of natural language that can be incrementally developed with further linguistic capabilities. Finally, it is our assumption that the system is used in "high value" contexts by participants who are willing and able to adapt to some of the constraints (topic, expressivity, explicit marking of statement relations) in order to gain the advantages of collaboratively building a clear, explicit knowledge base which represents a range of diverse, and possibly conflicting statements. Related issues arise with wide-scale parsers [17].

To generate instances of arguments and their relations, we require explicit indicators of contradictoriness. Consider statements such as *Bill exercises daily* and *Bill does not exercise daily*; the lexical items and syntactic form are such that it is relatively easy

[10] 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 [27]. 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.

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 [29]. 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 ¬P, and *Bill is a couch potato* is R, ¬[P ∧ R] is a constraint – P and R cannot both be true in a model. Yet, unlike the *logical* contradiction [P ∧ ¬P] of Propositional Logic, *non-logical* 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 explicitly express the rules that represent the implicit knowledge which gives rise to the inconsistency. Such rules may be encoded in auxiliary KBs, e.g. ontologies or lexical semantic structures. 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. More broadly, contradictoriness is a rich, complex, research domain that is only recently yielding results [29,30,31,12,32].

A final 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 [33]. A related issue is that some patterns of argumentation may contain enthymemes in addition to those used to identify contradictions [34,35].

On the construction of well-formed rules, 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 [36,37]; 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.

While all of these issues are substantial, 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), resulting in an *argument knowledge base* (AKB). The AKB is then input to an abstract argumentation framework to calculate extensions as 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 [4,38], which represents arguments in terms of classical logic (for related work see [39,40,41,42,4,43]). While we could discuss other approaches to instantiating arguments and relationships which use defeasible rules (e.g. [43,44]), we keep to the 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 explore an implemented example, we do not examine these issues further.

[4,38] 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 [38] 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 $\bot$ denotes inconsistency, $\Delta \vdash \bot$ 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 $\phi \nvdash \bot$. 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 *counterarguments*. $< \Psi, \beta >$ 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.[11] $< \Psi, \beta >$ 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 [38]): 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. [4,38] define and discuss a range of these relationships among arguments; however, additional definitions are not directly relevant to our key 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, [4] 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 [4]; given an LRKB, JArgue generates arguments, attacks between arguments, and an argument tree.[12] 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.

From Section 2.1, we have the propositions 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 household's garbage.

[11] 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.

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

p2: No household should pay some tax for the household's 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 [45]); 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.[13]

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

[13] This may be a more substantive issue since what arguments and relations are generated and used for input to the abstract framework may depend 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.

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:[14]

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

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. In effect, the argument is constructed by *backwards chaining* inference from the conclusion to the rules and propositions that justify it.

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:[15]

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

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

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

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

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

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)

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

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

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

More generally, every literal in the language 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 [36,37]).

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 considerations yield the following arguments:

a5: $< \{p18\}, p18 >$
a6: $< \{p14, !14|15\}, p15 >$

Such arguments may need to be manually introduced, generated from the underlying LRKB, or generated by contraposition, the latter 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)

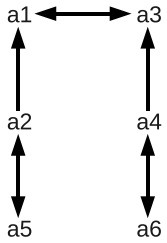

Fig. 2. Arguments and Relationships

The result is a graph as in Figure 2. 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. The argument 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 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? 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 is required to have an 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 input them to an abstract argumentation framework AF to calculate extensions. 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 [6], 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 [46,47,48], among others).

Definition 1. *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 2. *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 [5][16]; there are related tools that carry out similar computations [49,50,51] It is, then, very useful for our purposes, where we have created arguments in relations and then wish to

[16] 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`

compute extensions. ASPARTIX is implemented in the answer set programming language *Disjunctive Datalog System* (DLV).[17] 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 ASPARTIX 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)}. Preferred extensions, which are maximal sets of "consistent" arguments (and related propositions), correlate to the notion of *normative positions* of deontic logic [7], which are maximal sets of consistent deontic expressions. Therefore, we call such extensions *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:

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

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 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 household's 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 household's garbage.
p7: No household which receives a benefit which is paid by a council recycles the household's 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 $\wedge$ p13] $\rightarrow$!p18
r5: [p5 $\wedge$ p15] $\rightarrow$!p1
r6: [p6 $\wedge$ p7 $\wedge$ p8] $\rightarrow$ p5
r7: p14 $\rightarrow$ 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 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 [4,38,52,53,54,55,56], 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. Our approach is also distinct from related work on the integration of various argumentation

tools and technologies, such as the Argument Interchange Format, ArguBlogging, and OVA, which do not bear on the natural langauge issues, nor argument generation [57].

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, [58]). 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 [59,60,61]).

References

1. Lippi, M., Torroni, P.: Argumentation Mining: State of the Art and Emerging Trends. ACM Transactions on Internet Technology **0**(0) (2016) To appear
2. Macintosh, A.: Moving toward "intelligent" policy development. IEEE Intelligent Systems **24**(5) (2009) 79–82
3. Forsythe, D.E., Buchanan, B.G.: Knowledge acquisition for expert systems: some pitfalls and suggestions. In: Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993) 117–124
4. Besnard, P., Hunter, A.: Elements of Argumentation. MIT Press (2008)
5. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument and Computation **1**(2) (2008) 147–177
6. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77**(2) (1995) 321–358
7. Sergot, M.: Normative positions. In McNamara, Prakken, eds.: Norms, Logics, and Information Systems. IOS Press (1998) 289–308
8. Wyner, A., van Engers, T., Bahreini, K.: From policy-making statements to first-order logic. In Andersen, K.N., Francesconi, E., Grönlund, Å., van Engers, T.M., eds.: EGOVIS. Volume 6267 of Lecture Notes in Computer Science., Springer (2010) 47–61
9. Prabowo, R., Thelwall, M.: Sentiment analysis: A combined approach. Journal of Informetrics **3**(1) (2009) 143–157
10. Wyner, A., Mochales-Palau, R., Moens, M.F., Milward, D.: Approaches to text mining arguments from legal cases. In Francesconi, E., Montemagni, S., Peters, W., Tiscornia, D., eds.:

Semantic Processing of Legal Texts. Volume 6036 of Lecture Notes in Computer Science. Springer (2010) 60–79
11. Wyner, A., Peters, W.: On rule extraction from regulations. In Atkinson, K., ed.: Legal Knowledge and Information Systems - JURIX 2011: The Twenty-Fourth Annual Conference, IOS Press (2011) 113–122
12. Voorhees, E.M.: Contradictions and justifications: Extensions to the textual entailment task. In: Proceedings of ACL-08: HLT, Columbus, Ohio, Association for Computational Linguistics (June 2008) 63–71
13. Wyner, A., Schneider, J., Atkinson, K., Bench-Capon, T.: Semi-automated argumentative analysis of online product reviews. In: Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012). (2012) 43–50
14. Wyner, A., Schneider, J.: Arguing from a point of view. In: Proceedings of the First International Conference on Agreement Technologies, Dubrovnick, Croatia (October 2012) 153–167
15. Schneider, J., Wyner, A.: Identifying consumers' arguments in text. In Maynard, D., van Erp, M., Davis, B., eds.: Proceedings of the 1st Workshop on Semantic Web and Information Extraction (SWAIE 2012). CEUR Workshop Proceedings, Galway, Ireland (2012) 31–42
16. Bos, J.: Wide-coverage semantic analysis with boxer. In Bos, J., Delmonte, R., eds.: Semantics in Text Processing. STEP 2008 Conference Proceedings. Research in Computational Semantics, College Publications (2008) 277–286
17. Wyner, A., Bos, J., Basile, V., Quaresma, P.: An empirical approach to the semantic representation of law. In: Proceedings of 25th International Conference on Legal Knowledge and Information Systems (JURIX 2012), Amsterdam, The Netherlands, IOS Press (2012) 177–180
18. Saint-Dizier, P.: Processing natural language arguments with the <TextCoop> platform. Argument & Computation **3**(1) (2012) 49–82
19. Villalba, M., Saint-Dizier, P.: Some facets of argument mining for opinion analysis. In Verheij, B., Szeider, S., Woltran, S., eds.: Proceeedings of the conference on Computational Models of Argumentation (COMMA 2012). (2012) 23–34
20. de Paiva, V., Bobrow, D.G., Condoravdi, C., Crouch, D., King, T.H., Karttunen, L., Nairn, R., Zaenen, A.: Textual inference logic: Take two. In: C&O:RR. (2007)
21. Cabrio, E., Villata, S.: Natural language arguments: A combined approach. In Raedt, L.D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F., eds.: ECAI. Volume 242 of Frontiers in Artificial Intelligence and Applications., IOS Press (2012) 205–210
22. Cabrio, E., Villata, S.: Generating abstract arguments: A natural language approach. [62] 454–461
23. Wyner, A., van Engers, T.: A framework for enriched, controlled on-line discussion forums for e-government policy-making. In Chappelet, J.L., Glassey, O., Janssen, M., Macintosh, A., Scholl, J., Tambouris, E., Wimmer, M., eds.: Electronic Government and Electronic Participation, Linz, Austria, Trauner Verlag (2010) 357–364
24. Wyner, A., Angelov, K., Barzdins, G., Damljanovic, D., Davis, B., Fuchs, N., Hoefler, S., Jones, K., Kaljurand, K., Kuhn, T., Luts, M., Pool, J., Rosner, M., Schwitter, R., Sowa, J.: On controlled natural languages: properties and prospects. In: Proceedings of the 2009 conference on Controlled natural language. CNL'09, Berlin, Heidelberg, Springer-Verlag (2010) 281–289
25. Besnard, P., Hunter, A.: Practical first-order argumentation. In Veloso, M.M., Kambhampati, S., eds.: AAAI, AAAI Press / The MIT Press (2005) 590–595
26. Efstathiou, V., Hunter, A.: An algorithm for generating arguments in classical predicate logic. In: ECSQARU. (2009) 119–130
27. Barwise, J., Cooper, R.: Generalized quantifiers and natural language. Linguistics and Philosophy **4** (1981) 159–219

28. Kuhn, T.: Controlled English for Knowledge Representation. PhD thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich (2010)
29. Cruse, A.: Lexical Semantics. Cambridge University Press (1986)
30. Bryne, E., Hunter, A.: Man bites dog: Looking for interesting inconsistencies in structured news reports. Data and Knowledge Engineering **48**(3) (2004) 265–295
31. de Marneffe, M.C., Rafferty, A.N., Manning, C.D.: Finding contradictions in text. In: Proceedings of ACL-08: HLT, Columbus, Ohio, Association for Computational Linguistics (June 2008) 1039–1047
32. Basile, V., Bos, J., Evang, K., Venhuizen, N.: Negation detection with discourse representation structures. In: The First Joint Conference on Lexical and Computational Semantics (SEM 2012), Montreal, Canada (2012) 301–309
33. Walton, D.: Argumentation Schemes for Presumptive Reasoning. Erlbaum, Mahwah, N.J. (1996)
34. Black, E., Hunter, A.: Using enthymemes in an inquiry dialogue system. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'08), ACM Press (2008) 437–444
35. Walton, D.: The three bases for the enthymeme: A dialogical theory. Journal of Applied Logic **6**(3) (2008) 361–379
36. Edgington, D.: Conditionals. In Zalta, E., ed.: Stanford Encyclopedia of Philosophy. Stanford University (2006) http://plato.stanford.edu/entries/conditionals/.
37. Stenning, K., van Lambalgen, M.: A little logic goes a long way: basing experiment on semantic theory in the cognitive science of conditional reasoning. Cognitive Science **28**(4) (2004) 481–529
38. Besnard, P., Hunter, A.: Argumentation based on classical logic. In Rahwan, I., Simari, G., eds.: Argumentation in Artificial Intelligence. Springer (2009) 133–152
39. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of Applied Non-Classical Logics **7**(1) (1997)
40. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming **4**(1) (2004) 95–137
41. Governatori, G., Maher, M.J., Antoniu, G., Billington, D.: Argumentation semantics for defeasible logic. Journal of Logic and Computation **14**(5) (2004) 675–702
42. Amgoud, L., Caminada, M., Cayrol, C., Lagasquie, M.C., Prakken, H.: Towards a consensual formal model: inference part. Technical report, ASPIC project (2004) Deliverable D2.2: Draft Formal Semantics for Inference and Decision-Making.
43. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument and Computation **1**(2) (2010) 93–124
44. Gordon, T., Prakken, H., Walton, D.: The Carneades model of argument and burden of proof. Artificial Intelligence **171** (2007) 875–896
45. Wyner, A.: An ontology in OWL for legal case-based reasoning. Artificial Intelligence and Law **16**(4) (2008) 361–387
46. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence **93** (1997) 63–101
47. Bench-Capon, T.J.M.: Persuasion in practical argument using value-based argumentation frameworks. Journal of Logic and Computation **13**(3) (2003) 429–448
48. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artificial Intelligence **171**(5-6) (2007) 286–310
49. Snaith, M., Reed, C.: TOAST: Online ASPIC$^+$ implementation. [62] 509–510
50. Ellmauthler, S., Wallner, J.: Evaluating abstract dialectical frameworks with ASP. [62] 505–506

51. Brewka, G., Gordon, T.F.: Carneades and abstract dialectical frameworks: A reconstruction. In Baroni, P., Cerutti, F., Giacomin, M., Simari, G.R., eds.: COMMA. Volume 216 of Frontiers in Artificial Intelligence and Applications., IOS Press (2010) 3–12
52. Prakken, H.: Formalising ordinary legal disputes: a case study. Artificial Intelligence and Law **16** (2008) 333–359
53. Wyner, A., Bench-Capon, T., Atkinson, K.: Arguments, values and baseballs: Representation of Popov v. Hayashi. In Lodder, A.R., Mommers, L., eds.: Legal Knowledge and Information Systems. JURIX 2007, Amsterdam, IOS Press (2007) 151–160
54. Toni, F., Torroni, P.: Bottom-up argumentation. In: Proceedings of the First International Workshop on the Theory and Applications of Formal Argumentation (TAFA11). Volume 11., Barcelona, Spain (2011) 291–303
55. Wyner, A., Bench-Capon, T., Atkinson, K.: Towards formalising argumentation about legal cases. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Law (ICAIL 2011), Pittsburgh, PA, USA (2011) 1–10
56. Prakken, H.: Formalising a legal opinion on a legislative proposal in ASPIC+. In: Legal Knowledge Based Systems. JURIX 2012: The Twenty-Fifth Annual Conference, Amsterdam, IOS Press (2012) 119–128
57. Snaith, M., Devereux, J., Lawrence, J., Reed, C.: Pipelining argumentation technologies. In Baroni, P., Cerutti, F., Giacomin, M., Simari, G.R., eds.: Proceedings of the Third International Conference on Computational Models of Argument (COMMA 2010), IOS Press (2010) 447–454
58. Walton, D., Atkinson, K., Bench-Capon, T., Wyner, A., Cartwright, D.: Argumentation in the framework of deliberation dialogue. In Bjola, C., Kornprobst, M., eds.: Arguing Global Governance: Agency, Lifeworld and Shared Reasoning. Routledge (2010) 210–230
59. Mann, W.C., Thompson, S.A.: Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, University of Southern California, Information Sciences Institute (ISI) (June 1987)
60. Grasso, F.: Towards a framework for rhetorical argumentation. In Bos, J., Foster, M.E., Matheson, C., eds.: Proceedings of the 6th Workshop on the Semantics and Pragmatics of Dialogue (EDILOG-2002), Edinburgh, UK (2002) 53–60
61. Sporleder, C., Lascarides, A.: Using automatically labelled examples to classify rhetorical relations: An assessment. Natural Language Engineering **14**(3) (2006) 369–416
62. Verheij, B., Szeider, S., Woltran, S., eds.: Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012. In Verheij, B., Szeider, S., Woltran, S., eds.: COMMA. Volume 245 of Frontiers in Artificial Intelligence and Applications., IOS Press (2012)