Dispute Resolution with OWL DL and Reasoning

Ildar Baimuratov[1,2], Elena Lisanyuk[3,4] and Dmitry Prokudin[3]

[1]*L3S Research Center, Leibniz University Hannover, Hannover, Germany*
[2]*TIB - Leibniz Information Centre for Science and Technology, Hannover, Germany*
[3]*St Petersburg State University, St Petersburg, Russia*
[4]*National Research University Higher School of Economics, Moscow, Russia*

Abstract
Dispute resolution is an essential part of argumentation. Although there have been particular advances in computational argumentation with machine learning, we assume that explicit and deterministic techniques for dispute resolution still have significant potential. There are semantic approaches to argument representation, such as the Argument Interchange Format that supports a certain level of interoperability, but according to our knowledge, dispute resolution techniques based on semantics are not yet studied. In this research, we consider single mixed disputes through abstract argumentation frameworks and propose a method for representing such disputes in OWL DL that allows resolving them with reasoning. Additionally, we develop an algorithm for generating the OWL DL representation having argument sets and attack relations as input. The algorithm is validated with a proof of concept implementation, and the OWL DL representation – with an example of correct dispute resolution performed by reasoning over the generated ontology.

Keywords
Dispute resolution, abstract argumentation frameworks, computational argumentation, OWL DL, reasoning

1. Introduction

An argument is a piece of reasoning implying a transition from premises to a conclusion that expresses the opinion to be defended. Argumentation can be considered from different angles, among them are persuasion, proof, decision-making, information-seeking, dispute resolution, etc. Argumentation is a multidisciplinary research field that spans across diverse areas such as logic, philosophy, language, rhetoric, law, psychology, and computer science. Argumentation plays an essential role in artificial intelligence study, due to its ability to conjugate user-related cognitive models with computational models for automated reasoning. Neighboring areas inside the artificial intelligence domain include machine learning, natural language processing, discourse analysis, computational linguistics, information extraction, and knowledge representation. Computational argumentation has recently become a hot topic due to recent advances in machine learning methods that promise to enable applications of artificial intelligence to social, economic sciences, and policymaking.

DL 2023: 36th International Workshop on Description Logics, September 2–4, 2023, Rhodes, Greece
baimuratov.i@gmail.com (I. Baimuratov); e.lisanuk@spbu.ru (E. Lisanyuk); hogben.young@gmail.com (D. Prokudin)
0000-0002-6573-131X (I. Baimuratov); 0000-0003-0135-4583 (E. Lisanyuk); 0000-0002-9464-8371 (D. Prokudin)

Although there are particular advances in argumentation mining [1], we assume that explicit and deterministic techniques for dispute resolution still have significant potential. A trustworthy dispute resolution technique requires a representation format for dialogical argumentation that would be machine- and human-readable, and enable an algorithmic procedure to derive the solution. In this research, we aim at OWL-based argumentation representations that provide dispute resolution with reasoning.

Disputes are divided into single and multiple ones according to the number of propositions that constitute the parties' points of view, and into unmixed and mixed ones, depending on the parties' intentions to defend their point of view or criticize other opinions, or do both, respectively [2]. In this study, we consider only single mixed disputes.

The contributions of the present work are the following: 1) a method for representing argumentation with OWL DL[1] that is based on abstract argumentation frameworks and supports dispute resolution with reasoning, and 2) an algorithm for generating the OWL representation having argument sets and attack relations as input. The algorithm is validated with a proof of concept implementation, and the OWL DL representation – with an example of the correct dispute resolution performed by reasoning over the generated ontology.

2. Related Work

A method for the formal representation of an argument is called an argumentation framework. There are several argumentation frameworks. In [3], Phan Minh Dung proposed abstract argumentation frameworks, where an argument is an abstract entity whose role is determined solely by its relationship to other arguments. The former idea allows abstracting from the structural distinctions among them, making the notion of argument demonstration derivative from the notion of acceptability of arguments.

Considering the computational aspect, currently, the most widely used argument representation is the Argument Interchange Format (AIF) [4]. It is intended to exchange data between various argumentation tools and applications. Its abstract model includes three groups of entities: 1) argument entities and relations between them, 2) entities describing the communication of arguments, and 3) ones describing the context of arguments. Arguments are represented as directed graphs, called argumentation networks, where nodes represent statements or argumentation patterns, and edges are unlabeled. Various implementations of AIF were considered, including one in the RDF format[2]. In [5], an OWL DL implementation of AIF was proposed. It allows the classification and inference of argument schemes, but the dispute resolution problem and dialogical argumentation, in general, were not considered. The AIF format is used in the OVA tool [6] for analyzing and annotating argumentation texts in natural language. The annotation can be represented in formats such as JSON, RDF and Prolog. For the accumulation of annotated texts, an online database AIFdb was created [7].

A project [8] develops a software system designed to support the study of argumentation in popular science texts in the Russian language. For annotating texts, the authors extend AIF model with the weight of the argument scheme and of its elements. Data is represented in RDF

[1]https://www.w3.org/TR/owl-guide/
[2]https://www.w3.org/RDF/

format. The system implements the identification of argumentative indicators and automated evaluation of the persuasiveness of the arguments.

Another argument representation format, Argdown[3] is a Markdown[4]-inspired argument mark-up language, implemented as a context-free grammar and parser. It allows representing dialogical argumentation and generating argument maps but does not provide any dispute resolution procedures.

The legal domain is closely related to the argumentation theory. The Legal Knowledge Interchange Format (LKIF) [9] is designed for two purposes: 1) as a reusable and extensible core ontology for legal knowledge bases, and 2) as an interchange format for existing legal knowledge representation languages. The LKIF ontology includes entities for cases, obligations, and relations between them. Similarly to AIF, LKIF annotates argumentation rules as nodes of a specific argumentation scheme. LKIF rules are implemented not within OWL DL syntax, but as an extension of SWRL with support for negation and defeasible reasoning. LKIF is used via the XML format in the Carneades system [10].

In [11], the authors proposed an approach for the advanced access and reasoning facilities on legislation. The authors use the provision model that includes various provision types and attributes. They formalize it with OWL DL language and use it together with domain ontologies to form a structural mark-up of legal texts. They demonstrate how reasoning can be applied to infer indirect relations between provisions and how to query them with SPARQL. However, the model is too domain-specific and can not be scaled for argumentation.

3. Method

3.1. Abstract Argumentation Framework

Our dispute resolution algorithm is based on Dung's abstract argumentation framework, described in [3].

Definition 1. *An **argumentation framework** AF is a pair*

$$AF =< AR, attacks >,$$

where AR is a set of arguments and $attacks \subseteq AR \times AR$.

We say that an argument a attacks an argument b (or b is attacked by a) if $attacks(a, b)$ holds. Similarly, we say that a set S of arguments attacks a (or a is attacked by S) if a is attacked by an argument in S. The binary abstract relation $attacks(a, b)$ between arguments a and b symbolizes how the criticism of argument a rejects argument b; and how a further critical development in the dialog $attacks(c, a)$ rejects argument a by counterattacking it and thereby returns argument b as defended.

Unlike the entailment relation, the attack relation is not reflexive, symmetric, or transitive. This transferred the formal logical representation of argumentation from the microlevel of the internal structure of the argument to the macrolevel of the structure of argumentation in the

[3]https://argdown.org/
[4]https://daringfireball.net/projects/markdown/syntax

dispute and made it possible to overcome the restrictions of the classical argumentation analysis on modeling criticism.

Another feature of this argumentation framework is the usage of procedural semantics instead of classical logic frameworks. For a set of arguments, procedural semantics allows to define subsets with specific properties, including consistency and completeness, where required. Thus, Dung defines a conflict-free set of arguments.

Definition 2. *A set S of arguments is said to be **conflict-free** if there are no arguments a and b in S such that a attacks b.*

The notion of acceptability allows determining dispute outcomes.

Definition 3. *An argument $a \in AR$ is **acceptable** with respect to a set S of arguments iff for each argument $b \in AR$: if b attacks a then b is attacked by S.*

Finally, an admissible set of arguments is a set of acceptable arguments.

Definition 4. *A conflict-free set of arguments S is **admissible** iff each argument in S is acceptable with respect to S.*

For a resolution of the single unmixed dispute, it is sufficient that there is a nonempty admissible subset of arguments, that is, there is at least one acceptable argument. Thus, the dispute resolution problem can be considered as a classification of arguments into admissible sets.

Example 1. *Let us consider the original example of argumentation from Dung's paper [3]:*

- *I: My government cannot negotiate with your government because your government doesn't even recognize my government.*
- *A: Your government doesn't recognize my government either.*
- *I: But your government is a terrorist government.*

This argument is formalized as two argumentation sets $I = \{i_1, i_2\}$ and $A = \{a\}$, and attack pairs $\{(i_1, a), (a, i_1), (i_2, a)\}$. Both sets are conflict-free, but the argument a is not acceptable w.r.t. I, as it is attacked by i_2, but no argument from A attacks i_2. Thus, there is a non-empty admissible subset only for I.

3.2. OWL DL Implementation

Our OWL DL implementation of Dung's framework is designed in a way that provides the automatic classification of arguments into admissible sets with reasoning. We illustrate the implementation with listings in the Manchester syntax.

The formalization of the basic elements of the framework is straightforward. Each argument set is represented as an owl:Class.

```
Class: <onto.owl#A>

    SubClassOf:
        owl:Thing
```

Listing 1: Example of an argument set

Each argument is considered to be an owl:NamedIndividual. The belonging of an argument to the argument set is represented with the rdf:type relation. The text of the argument is stored in rdfs:label.

```
Individual: <onto.owl#a>

    Annotations:
        rdfs:label "Your government doesn't recognize my government either"^^xsd:string

    Types:
        <onto.owl#A>
```

Listing 2: Example of an argument

To represent the *attacks* relation, the owl:ObjectProperty 'attacks' is defined (for clearness, here and further we write custom OWL entities in apostrophes). Also, the property 'isAttackedBy' is defined as owl:inverseOf 'attacks'. It is required for defining admissible sets of arguments.

```
ObjectProperty: <onto.owl#attacks>

    InverseOf:
        <onto.owl#isAttackedBy>
```

Listing 3: Declaration of the 'attacks' relation

Now we can assert that one argument attacks another.

```
Individual: <onto.owl#a>
...
    Facts:
     <onto.owl#attacks>  <onto.owl#i1>
```

Listing 4: Example of asserting the 'attacks' relation

In order to provide reasoning under Open World Assumption (OWA), we have to "close" each individual argument with respect to the list of arguments it attacks. We do this by asserting that the argument belongs to an unnamed class that has the relation 'attacks' to the set of arguments it attacks, constructed with owl:oneOf operator under owl:allValuesFrom restriction. If the argument attacks no argument, we assert that it attacks only owl:Nothing.

```
Individual: <onto.owl#a>
...
    Types:
...
        <onto.owl#attacks> only ({onto.owl#i1>}),
...
```

Listing 5: Example of closing an argument regarding the 'attacks' relation

For the same reason, we also close each individual argument regarding the relation 'isAttackedBy'.

```
Individual: <onto.owl#a>
...
    Types:
...
        <onto.owl#isAttackedBy> only ({<onto.owl#i1> , <onto.owl#i2>})
...
```

Listing 6: Example of closing an argument regarding the 'isAttackedBy' relation

To define a conflict-free set in a manner that supports reasoning under the OWA, we can not use the owl:complementOf operator. Thus, for each argument set we form a union of all other argument sets with owl:unionOf. Then its conflict-free subset is defined as a class that has the relation 'attacks' only (owl:allValuesFrom) to the union of other argument sets.

```
Class: <onto.owl#AConflictFree>

    EquivalentTo:
        <onto.owl#A>
         and (<onto.owl#attacks> only (<onto.owl#I>))

    SubClassOf:
        <onto.owl#A>
```

Listing 7: Example of a conflict-free set

For each conflict-free set, an admissible subset is defined as a class that has the relation 'isAttackedBy' only (owl:allValuesFrom) to the arguments that have the relation 'isAttackedBy' to some (owl:someValuesFrom) arguments from the initial conflict-free set. Thus, if an argument is attacked by no arguments, it also belongs to the admissible set.

```
Class: <onto.owl#AAdmissible>

    EquivalentTo:
        <onto.owl#AConflictFree>
         and (<onto.owl#isAttackedBy> only (<onto.owl#isAttackedBy> some <onto.owl#
             AConflictFree>))

    SubClassOf:
        <onto.owl#AConflictFree>
```

Listing 8: Example of an admissible set

Example 2. *Let us implement the argument from the Example 1 in OWL DL. The resulting class hierarchy visualized with Protege interface is presented in Fig. 1. The full declarations of the arguments i_1, a and i_2 are presented in Listings 9, 10 and 11 respectively.*

```
Individual: <onto.owl#i1>

    Annotations:
        rdfs:label "My government cannot negotiate with your government because your
            government doesn't even recognize my government"^^xsd:string
```

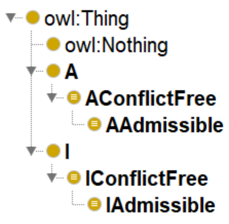

Figure 1: The hierarchy of classes for Example 1

```
    Types:
        <onto.owl#I>,
        <onto.owl#attacks> only ({<onto.owl#a>}),
        <onto.owl#isAttackedBy> only ({<onto.owl#a>})

    Facts:
     <onto.owl#attacks>  <onto.owl#a>
```

Listing 9: Declaration of i_1

```
Individual: <onto.owl#a>

    Annotations:
        rdfs:label "Your government doesn’t recognize my government either"^^xsd:string

    Types:
        <onto.owl#A>,
        <onto.owl#attacks> only ({<onto.owl#i1>}),
        <onto.owl#isAttackedBy> only ({<onto.owl#i1> , <onto.owl#i2>})

    Facts:
     <onto.owl#attacks>  <onto.owl#i1>
```

Listing 10: Declaration of a

```
Individual: <onto.owl#i2>

    Annotations:
        rdfs:label "But your government is a terrorist government"^^xsd:string

    Types:
        <onto.owl#I>,
        <onto.owl#attacks> only ({<onto.owl#a>}),
        <onto.owl#isAttackedBy> only owl:Nothing

    Facts:
     <onto.owl#attacks>  <onto.owl#a>
```

Listing 11: Declaration of i_2

3.3. Ontology Generating Algorithm

The algorithm is intended to generate the described OWL DL representations of arguments using the list of argument sets and attack pairs as input. For better readability, the algorithm is divided into three steps.

Algorithm 1 declares basic entities including argument sets, arguments, 'attacks' and 'isAttackedBy' relations, and infers 'isAttackedBy' statements with reasoning. Not considering reasoning, the algorithm has linear complexity, as it only runs once through both lists of arguments in argument sets and of attack pairs.

Algorithm 1 Basic entities generation

```
Require: Argument sets, attack pairs
  Create ontology $O$
  for argument set $A$ in argument sets do
      Create an owl:Class $C$
      for argument $a \in A$ do
          Create an owl:NamedIndividual $i$ of rdf:type $C$
      end for
  end for
  Create owl:ObjectProperty 'attacks'
  Create owl:ObjectProperty 'isAttackedBy'
  Assert 'isAttackedBy' owl:inverseOf 'attacks'
  for $(a_1, a_2)$ in attack pairs do
      Find the instance $i_1$ in $O$ for $a_1$
      Find the instance $i_2$ in $O$ for $a_2$
      Assert $i_1$ 'attacks' $i_2$
  end for
  Run reasoner
```

Algorithm 2 closes arguments regarding 'attacks' and 'isAttackedBy' relation to support the reasoning under OWA. This algorithm also has linear complexity regarding the number of arguments.

Algorithm 3 declares conflict-free and admissible subsets of arguments for each input argument set and classifies arguments into them with reasoning. As declaring a conflict-free set for the argument set requires constructing a list of other argument sets, this algorithm has quadratic complexity regarding the number of argument sets.

Thus, the algorithm generates an ontology that contains an empty admissible subset for each argument set. These admissible subsets are then populated by classifying arguments with reasoning.

4. Proof of Concept

Input Data For proof of concept, the input data, i.e. argument sets and attack pairs, are represented in JSON format. Its structure includes two high-level keys: *argument_sets* and

Algorithm 2 Closing world

```
Require: Ontology $O$
  for argument instance $i$ in $O$ do
     Get the set $Attacks_i$ of the arguments $i$ attacks
     if $Attacks_i$ is empty then
        Assert $i$ rdf:type ('attacks' owl:AllValuesFrom owl:Nothing)
     else
        Assert $i$ rdf:type 'attacks' owl:AllValuesFrom owl:OneOf($Attacks_i$))
     end if
     Get the set $IsAttackedBy_i$ of the arguments $i$ is attacked by
     if $IsAttackedBy_i$ is empty then
        Assert $i$ rdf:type ('isAttackedB' owl:AllValuesFrom owl:Nothing)
     else
        Assert $i$ rdf:type ('isAttackedBy' owl:AllValuesFrom owl:OneOf($IsAttackedBy_i$))
     end if
  end for
```

Algorithm 3 Inferring solution

```
Require: Argument sets, ontology $O$
  for argument set $A_i$ in argument sets do
     Find the class $C_i$ for $A_i$ in $O$
     Create its subclass $C_{cf}$
     Create an empty set $S$
     for argument set $A_j$ in argument sets do
        if $A_i$ is not $A_j$ then
           Find the class $C_j$ for $A_j$ in $O$
           Add $C_j$ to $S$
        end if
     end for
     Assert $C_{cf}$ owl:equivalentClass (owl:intersectionOf ($C_i$, 'attacks' owl:allValuesFrom
  owl:unionOf($S$)))
     Create subclass $C_a$ of $C_{cf}$
     Assert $C_a$ owl:equivalentClass (owl:intersectionOf ($C_{cf}$, 'isAttackedBy'
  owl:allValuesFrom ('isAttackedBy' owl:someValuesFrom $C_{cf}$)))
  end for
  Run reasoner
```

attack_pairs. The key *argument_sets* has a dictionary, where each item represents an argument set. The argument set item has a label and a dictionary of arguments that belong to it. Each argument item in the latter dictionary has a label and a text. The key *attack_pairs* has a list of two-element lists of argument labels that represent attack pairs. Formatted input data for Example 1 are provided in Listing 12.

```
{
  "argument_sets": {
    "I": {
      "i1": "My government cannot negotiate with your government because your government
          doesn’t even recognize my government",
      "i2": "But your government is a terrorist government"
    },
    "A": {
      "a": "Your government doesn’t recognize my government either"
    }
  },
  "attack_pairs": [
    [
      "i1", "a"
    ],
    [
      "a", "i1"
    ],
    [
      "i2", "a"
    ]
  ]
}
```

Listing 12: Input data based on Example 1

Prototype The algorithm was implemented with Python language using Owlready2 library[5] for working with OWL and the Pellet reasoner [12]. The prototype is available in GitHub[6], as well as the input data and the ontology generated from Example 1. Additionally, a separate script runs the reasoner over the generated ontology to classify arguments into conflict-free or admissible sets.

Results The most intransparent parts of the algorithm are running reasoner for 1) inferring ’isAttackedBy’ assertions, and 2) classifying arguments into conflict-free and admissible sets. We provide the logs of these runs, conducted in Windows 10 OS with Intel Core i7-11850H CPU and 16GB RAM. The first run took 0.72 s. and as a result, 6 individual assertions were inferred, see Listing 13. The second run classified 3 individuals for 0.75 s., see Listing 14.

```
* Owlready2 * Pellet took 0.7212517261505127 seconds
* Owlready2 * Pellet output:

 http://www.w3.org/2002/07/owl#Thing
    onto.owl#A - (onto.owl#a)
    onto.owl#I - (onto.owl#i1, onto.owl#i2)

PROPINST: onto.owl#i1 onto.owl#attacks onto.owl#a
PROPINST: onto.owl#i1 onto.owl#isAttackedBy onto.owl#a
PROPINST: onto.owl#a onto.owl#attacks onto.owl#i1
PROPINST: onto.owl#a onto.owl#isAttackedBy onto.owl#i1
```

[5]https://owlready2.readthedocs.io/en/latest/
[6]https://github.com/ldrbmrtv/computational_argumentation_dung

```
PROPINST: onto.owl#a onto.owl#isAttackedBy onto.owl#i2
PROPINST: onto.owl#i2 onto.owl#attacks onto.owl#a
```

Listing 13: Inferring 'isAttackedBy' relation

```
* Owlready2 * Pellet took 0.75439453125 seconds
* Owlready2 * Pellet output:

 http://www.w3.org/2002/07/owl#Thing
    onto.owl#A
       onto.owl#AConflictFree - (onto.owl#a)
          onto.owl#AAdmissible
    onto.owl#I
       onto.owl#IConflictFree
          onto.owl#IAdmissible - (onto.owl#i1, onto.owl#i2)
```

Listing 14: Classifying arguments

In result, the arguments i_1 and i_2 were correctly classified as acceptable, and assigned to the class $IAdmissible$, i.e. the admissible subclass of the argument set I. And the argument a was correctly assigned only to the class $AConflictFree$, i.e. the conflict-free subset of the argument set A.

5. Conclusion

In this research, we reviewed existing solutions related to semantic argument representation and concluded that among them, there is no approach that would provide a dispute resolution technique. Based on Dung's abstract argumentation frameworks, we proposed a method for OWL DL representation of single mixed disputes designed in a manner that allows their resolution with reasoning. Moreover, we proposed an algorithm for the generation of the OWL DL representations having argument sets and attack pairs as input. The algorithm has quadratic complexity and is validated with a proof of concept implementation. The proposed representation is validated with an example of the correct dispute resolution over the generated ontology. Future work will address extending the present approach for implementing complete, preferred and stable extensions of argument sets, and developing approaches for the computational resolution of other types of disputes. The latter may require considering other argumentation frameworks. We also plan to automate the input data preparation with text annotation, NLP techniques and machine learning.

Acknowledgments

This work was co-funded by the European Research Council for the project ScienceGRAPH (Grant agreement ID: 819536) and NFDI4DataScience funded by the German Research Foundation (DFG project number 460234259). The support from the Russian Science Foundation, project No. 20-18-00158, realised at St. Petersburg State University, is kindly recognized.

References

[1] M. Lippi, P. Torroni, Argumentation mining: State of the art and emerging trends, ACM Trans. Internet Technol. 16 (2016). doi:10.1145/2850417.

[2] F. H. Van Eemeren, R. Grootendorst, A systematic theory of argumentation: The pragma-dialectical approach, Cambridge University Press, 2004.

[3] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (1995) 321–357. doi:10.1016/0004-3702(94)00041-X.

[4] C. Chesñevar, Mcginnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, S. Willmott, Towards an argument interchange format, The Knowledge Engineering Review 21 (2006) 293–316. doi:10.1017/S0269888906001044.

[5] I. Rahwan, C. Reed, The argument interchange format, Argumentation in Artificial Intelligence (2009) 383–402. doi:10.1007/978-0-387-98197-0_19.

[6] M. Janier, J. Lawrence, C. Reed, Ova+: An argument analysis interface, Frontiers in Artificial Intelligence and Applications 266 (2014) 463–464. doi:10.3233/978-1-61499-436-7-463.

[7] J. Lawrence, F. Bex, C. Reed, M. Snaith, Aifdb: Infrastructure for the argument web, in: Computational Models of Argument, IOS Press, 2012, pp. 515–516.

[8] Y. Zagorulko, N. Garanina, A. Sery, O. Domanov, Ontology-based approach to organizing the support for the analysis of argumentation in popular science discourse, in: Artificial Intelligence: 17th Russian Conference, RCAI 2019, Ulyanovsk, Russia, October 21–25, 2019, Proceedings 17, Springer, 2019, pp. 348–362.

[9] A. Boer, R. Winkels, F. Vitali, MetaLex XML and the Legal Knowledge Interchange Format, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 21–41.

[10] T. F. Gordon, Introducing the carneades web application, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law, 2013, pp. 243–244.

[11] E. Francesconi, Semantic model for legal resources: Annotation and reasoning over normative provisions, Semantic Web 7 (2016) 255–265. doi:10.3233/SW-140150.

[12] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical owl-dl reasoner, Web Semant. 5 (2007) 51–53. doi:10.1016/j.websem.2007.03.004.