Introducing ALIAS

Simon Wells[1] and Roberto La Greca[2]

[1] Edinburgh Napier University s.wells@napier.ac.uk
[2] Edinburgh Napier University roberto@robertolagreca.com

Abstract. ALIAS is a Python library for constructing, manipulating, storing, visualising, and converting argumentation structues. It is available with full source code under a copyleft license and aims to become a *Swiss Army Knife* for working with arguments in a variety of end-user, researcher, pedagogical, and developer contexts.

1 Introduction

This paper introduces "A Library for Implementing Argumentation Systems" (ALIAS)[3], a software library for describing and working with arguments. In the first instance ALIAS supports Dung-style Abstract Argumentation Frameworks (AAFs) [1]. AAFs and their associated semantic machinery, such as extensions and labellings, and additional enhancements such as extensions, preferences, bipolarity, etc. have become a popular approach to defeasibly reasoning over suitably structured collections of argumentative data. Whilst AAFs often appear divorced from natural argument, seeming to concentrate on increasingly esoteric abstractions, Bench-Capon recently presented a good overview of how developments within the domain of AAFs have contributed to an increased understanding of and capacity for reasoning about arguments within the Artificial Intelligence and Law domain [2]. It is in this spirit, that natural language arguments can be abstracted into AAFs and that the resulting frameworks can be *usefully* analysed using abstract techniques, that this paper is presented. The remainder of this paper is structured as follows: in section 2 we introduce the ALIAS library and briefly discuss our motivation; in sections 3 and 4 we briefly discuss two of the contexts in which ALIAS can be used; in section 5 we discuss some additional features of the tool; finally in section 6 we draw some conclusions and outline directions for future work.

2 Introducing ALIAS

There were a number of motivations for the development of ALIAS. Amongst these are the lack of free and open libraries for working with arguments, the lack of straightforward and naive implementations of AAF algorithms, for example that demonstrate

[3] ALIAS is a free and open source project whose community page is hosted at GitHub https://github.com/alias-org/alias

the process of calculating a labelling or extension in a context useful for pedagogical purposes. Most importantly are the lack of flexible and extensible standalone tools for working with arguments in a research context and the need for tools that enable students to explore argumentss within a pedagogical context. Whilst there are many argument tools, many have drawbacks such as not being available with source under a copyleft license, being implemented in none-mainstream languages that put the tools out of reach of many developers, or being hosted tools, running on third party web-servers and thus leaving the end-user at the mercy of their tools host. ALIAS aims to provide an exemplary toolkit, applicable within research and pedagogical roles, available with source under a community-oriented copy-left licence, flexibly deployable in interactive, standalone, library, pipelined, and hosted contexts, implemented in a popular mainstream language that serves both software and web development communities, and designed with sufficient flexibility and extensibility to become a useful argumentation tool in a wide variety of contexts. Currently ALIAS has three core contexts of use; as a standalone tool, as a programming library, and as an interactive pedagogic and research environment. At its simplest, ALIAS is a pure Python[4] library for defining and working with arguments and provides a Python native API for constructing an AAF, by adding arguments and attacks. Once constructed the AAF can be inspected to calculate semantics according to a variety of extensions (complete, preferred, stable) and labelling (complete, grounded, preferred, stable, semi-stable) based approaches. ALIAS also provides support for a range of input/output formats including Aspartix "apx"[3], trivial graph format "tgf"[5], dot language[6], JavaScript Object Notation (JSON)[7], and networkx[8] formats and utility functions (argument existence, get attackers, conflict-free, admissible, complete). This enables arguments to be imported or exported by a variety of third-party tools with ALIAS playing the role of interlingua and converting between the native formats of each tool as required. As a result when operating in the standalone context ALIAS can perform an important role in enabling argument data to be pipelined between diverse specialist tools. In the following two examples, the standalone and interactive environment contexts are explored but the use of ALIAS as a Python library is not shown, preferring instead to concentrate ALIAS as an argumentation tool rather than as a development library. However such usage is a straightforward and standard procedure that will be familar to the majority of Python developers.

3 Example: Using ALIAS in the Shell

ALIAS may be used in the Python shell, or else at the CLI by writing Python scripts. It is within the scripting context that ALIAS can work within a heterogeneous pipeline of tools. Both approaches make use of the same simple API to interact with the library. In the following example, the library is imported, a new framework object is created and given the name 'example'. Subsequently two arguments are added to the framework

[4] `https://www.python.org/`
[5] `https://en.wikipedia.org/wiki/Trivial_Graph_Format`
[6] `http://www.graphviz.org/content/dot-language`
[7] `http://json.org/`
[8] `https://networkx.github.io/`

using a list containing two elements ‘a’ and ‘b’ then an attack is added between the arguments.

Listing 1.1. Simple example of using ALIAS in the default Python Shell

```
>>> import alias
>>> framework = alias.ArgumentationFramework('example')
>>> framework.add_argument(['a','b'])
>>> framework.add_attack(('a','b'))
```

4 Example: Using ALIAS within iPython Notebooks

In this context, ALIAS is an interactive tool for learning about and experimenting with arguments. All of the features of ALIAS are available, as they would be in the shell, a script, or a programming context, however an iPython notebook[9] provides an environment for interspersing text, images, and code to form a dynamic and interactive notebook in which data can be manipulated on the fly, results gathered, interspersed with explanatory text. Such an approach is becoming increasingly popular not only within pedagogical contexts, for example, as a way to provide interactive exercises to students, but also within research, particularly in data-oriented physical sciences where it is taking on the role of interactive digital laboratory notebook and data exploration tool. An example of an iPython notebook showing the construction of an AAF interspersed with explanatory text can be seen in Figure 1.

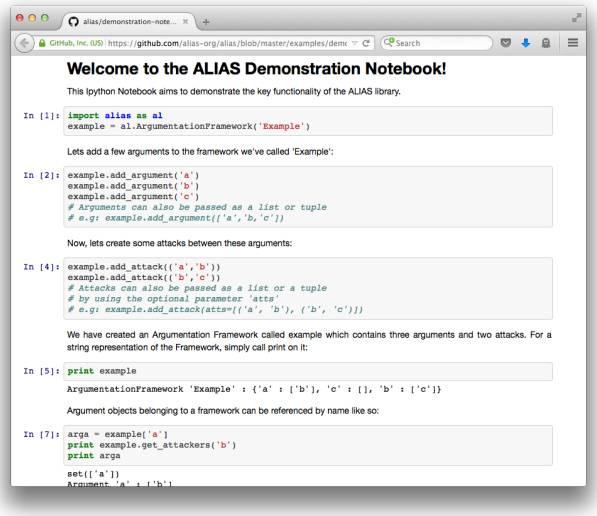

Fig. 1. The first several clauses of an iPython notebook using ALIAS

[9] `http://ipython.org/notebook.html`

5 Additional Features of ALIAS

With the aim of becoming a 'Swiss Army Knife' for Argumentation, ALIAS provides basic support for persistence of arguments and visualisation. Whilst early development has concentrated on AAFs the goal is to support representation and storage of both abstract and natural arguments alongside tooling for describing, storing, and manipulating both. In this respect ALIAS shares a similar space to Carneades [4] but rather than promulgating a specific approach, like Carneades structured argumentation, ALIAS ultimate aim is to provide tools for manipulating, transforming, and storing arguments in order to bridge between more specialist tools. SQLalchemy[10] is used to provide a database abstraction layer. By taking this approach, ALIAS incorporates a simple, consistent and extensible method for working with a range of underlying datastores. Because the choice of underlying datastore can not only affect the performance or scalability of a system but can also be problematic depending upon the skillset of users ALIAS does not prescribe a particular datastore. Data is persisted in the first instance within plain-text files on the filesystem using the range of formats outlined earlier. Additionally, support for persisting argument structures in the graph datastore Neo4J[11] is supported by default. Out of the box support for additional datastores is in the development roadmap for ALIAS and will extend to at least CouchDB[12] and MongoDB[13], using the aforementioned JSON serialisation functionality, and SQLite[14]. There are currently two options for visualisation of argument structures. The first option uses the network rendering facilities from the networkx library as illustrated in Figure 2. The second provides a standalone Javascript widget using the d3.js[15] Javascript visualisation library as illustrated in Figure 3 which shows the widget displaying within a browser window. The Javascript widget and a graph.json file can thus be exported from ALIAS and hosted on a webserver to enable dynamic visualisations to be deployed on the web. It should be noted that both visualisation approaches can also be embedded within iPython notebooks to provide in-line visualisations of arguments alongside the code that is generating and manipulating them.

6 Conclusions

ALIAS is in early but active development and there are a range of directions in which this work will be enhanced and proceed. Most importantly it will be deployment as a teaching aid and evaluated by students. A secondary thread is further development, on the one hand to increase the range of features, for example by incorporating support for calculating a greater diversity of labellings and extensions and on the other hand to make the system more performant. We plan to incorporate optional modules that enable semantics to be calculated using a range of techniques to enable both increased

[10] `http://www.sqlalchemy.org/`
[11] `http://neo4j.com/`
[12] `http://couchdb.apache.org/`
[13] `https://www.mongodb.org/`
[14] `https://www.sqlite.org/`
[15] `http://d3js.org/`

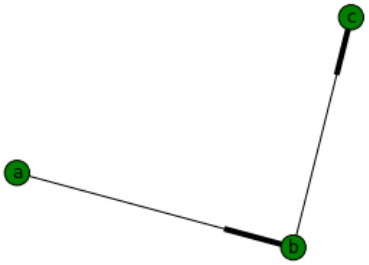

Fig. 2. A simple, 3 node AAF visualised using the networkx library native graph visualisation

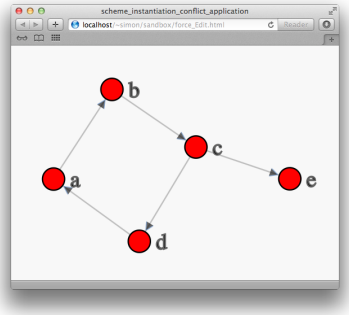

Fig. 3. A simple, 5 node AAF visualised using the ALIAS native javascript visualisation widget

performance and scalability. For example, implementing modules that use Answer Set Programming and/or Constraint Programming, using high performance libraries such as NumPy and SciPy, and using techniques such as parallelisation or memoization. An additional extension will also add support for the representation and manipulation of Concrete Argumentation Systems containing expressions in natural language. The first methods planned will enable representation of annotated natural language texts to be stored, perhaps taking Gate files as input [5], and thence for AAFs to be automatically derived from them. One goal is to support a pipeline from natural language through, mark up, analysis, manipulation, storage and visualisation, whilst providing a unified and stable API. To summarise, we have introduced ALIAS, described its current feature set, outlined plans to extend that feature set, and given some examples of usage constructing and manipulating simple argumentation frameworks.

References

1. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and *n*-person games. Artificial Intelligence **77** (1995) 321–357
2. Bench-Capon, T.: Open texture and argumentation: what makes an argument persuasive? In: Logic Programs, Norms and Action. Springer Berlin Heidelberg (2015) 220–233
3. Egly, U., Gaggl, S., Woltran, S.: Aspartix: Implementing argumentation frameworks using answer-set programming. In: Proceedings of the Twenty-Fourth International Conference on Logic Programming, (ICLP'08), Springer (2008)
4. Gordon, T.F., Prakken, H., Walton, D.: The carneades model of argument and burden of proof. Artificial Intelligence **171**(10–11) (2007) 875–896
5. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M.A., Saggion, H., Petrak, J., Li, Y., Peters, W.: Text Processing with GATE (Version 6). GATE (2011)