Towards DiArg: An Argumentation-based Dialogue Reasoning Engine[1]

Timotheus KAMPIK
Umeå University, Sweden

Dov GABBAY
University of Luxembourg, Luxembourg

Abstract. This paper presents ongoing work on the implementation of the *DiArg* argumentation-based reasoning engine that focuses on automating sequential argumentation for inquiry and deliberation dialogues. The engine uses abstract argumentation in its core and implements a meta-layer to support argument *context* and enforce the consistency of inferences in compliance with the cautious monotony and reference independence principles. In addition, DiArg can enforce *expansion* properties of an argumentation framework w.r.t. its predecessors in an argumentation framework sequence.

Keywords. Formal argumentation, Dialogue systems, Non-monotonic reasoning

1. Introduction

Formal argumentation has emerged as a promising line of research in the domain of artificial intelligence. In particular, a large body of theoretical research exists that can serve as the foundation for building knowledge-based systems. Indeed, recent research results demonstrate the competitiveness of argumentation-based approaches, for example for implementing explainable recommender systems [1]. However, relatively few software artifacts that provide reusable abstractions (in the form of software libraries with well-documented application programming interfaces) on formal argumentation approaches exist. One notable exception is the *Tweety Project*, which, among other features, provides Java libraries to define and resolve different types of formal argumentation frameworks [2]. This paper presents ongoing work on *DiArg*, which uses Tweety as a foundation to implement argumentation-based dialogue systems[2]. DiArg focuses on sequential argumentation, *i.e.*, the iterative resolution of sequences of argumentation frameworks, as well as on inquiry and deliberation aspects of dialogues, and can hence be considered a useful complement to argumentation-based dialogue reasoners that focus on strategic (game theoretical) aspects [3,4].

[1]We thank Juan Carlos Nieves. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

[2]In particular, Tweety provides abstractions and solvers for abstract argumentation frameworks to DiArg.

The rest of the paper is organized as follows. Section 2 provides an intuition of the most important theoretical preliminaries. Subsequently, Section 3 describes the DiArg engine and its implementation. Finally, Section 4 concludes the paper by highlighting limitations and outlining future work.

2. DiArg's Theoretical Foundations - A Semi-Formal Overview

This section provides an intuition of DiArg's theoretical foundations, some of which are work-in-progress (see, Kampik and Nieves [5,6]). At its core, DiArg resolves abstract argumentation frameworks. An abstract Argumentation Framework (AF) is a tuple $AF = (AR, AT)$, where AR is a set of propositional atoms and AT is a set of binary relations on $AR \times AR$ [7]. For $a, b \in AR$ such that $(a,b) \in AT$, we say that "a attacks b". $S \subseteq AR$ is conflict-free iff $\nexists a, b \in S$, such that "a attacks b". A key question in abstract argumentation is which sets of arguments in an AF can be considered "valid". Such a set of arguments is called an *extension*, and the function that determines the extensions of an AF is called a *semantics*. In this paper, we use *stage semantics* [8] as an example[3]. Given an argumentation framework $AF = (AR, AT)$, $S \subseteq AR$ is a stage extension of AF iff $S \cup S^+$ is maximal w.r.t. set inclusion among conflict-free sets in AR, where S^+ denotes all arguments that are attacked by any argument in S. $\sigma_{stage}(AF)$ denotes all stage extensions of AF. In DiArg dialogues, an initial argumentation framework is resolved (*i.e.*, its extensions are determined and one extension is selected as the AF's *conclusion*, either automatically or manually by a human user) and then iteratively *expanded* by adding new arguments and attack relations to it (and again resolved, and so forth). In this context, we distinguish between different types of *expansions* [9]:

- An argumentation framework $AF' = (AR', AT')$ is an expansion of an argumentation framework $AF = (AR, AT)$ (denoted by $AF \preceq_E AF'$) iff $AR \subseteq AR'$ and $AT \subseteq AT'$.
- An argumentation framework $AF' = (AR', AT')$ is a normal expansion of an argumentation framework $AF = (AR, AT)$ (denoted by $AF \preceq_N AF'$) iff $AF \preceq_E AF'$ and $(AR \times AR) \cap (AT' \setminus AT) = \{\}$.

Colloquially speaking, an AF's expansion "adds" new arguments and/or attack relations to the AF, but "removes" neither existing arguments nor existing attack relations. A normal expansion is an expansion that neither removes nor adds attack relations between existing arguments. Analogously, iff an argumentation framework AF' is an expansion of an argumentation framework AF, we call AF a *submodule* of AF', and iff AF' is a normal expansion of AF, we call AF a *normal submodule* of AF'.

When iteratively adding arguments and attack relations to an AF, DiArg creates *Argumentation Framework sequences* (AF sequences) that can be configured to be expanding or normally expanding. An AF sequence is a sequence of argumentation frameworks $AFS = < AF_0, ..., AF_n >$; the sequence is *expanding* iff each $AF_i \in AFS, 0 < i \leq n$ is an expansion of its predecessor AF_{i-1}, and *normally expanding* iff each $AF_i \in AFS, 0 < i \leq n$ is a normal expansion of its predecessor AF_{i-1}. After DiArg has determined the extensions

[3] Let us note that we use stage semantics primarily because it is convenient for demonstration purposes. Through the Tweety library, the DiArg reasoner supports a range of admissible set-based, as well as maximal conflict-free set-based argumentation semantics.

of an AF in an AF sequence, one extension is selected and logged as the AF's *conclusion*, either automatically by DiArg, or by a user. In any case, DiArg can ensure the conclusion is valid according to the configured argumentation semantics and other constraints.

DiArg can enforce that the conclusions that are derived from an AF sequence are aligned with the following principles[4]:

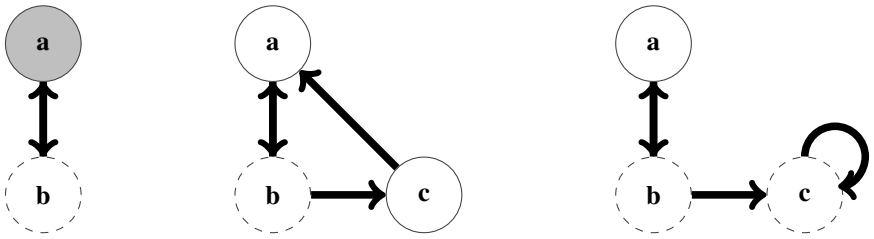
- **Reference independence** Given two argumentation frameworks $AF = (AR, AT)$ and $AF' = (AR', AT')$, such that AF' is a normal expansion of AF, and given the conclusions $A \subseteq AR$ from AF and $A' \subseteq AR'$ from AF', it holds true that $A' \not\subseteq AR \vee A' = A$.
- **Cautious monotony** Given two argumentation frameworks $AF = (AR, AT)$ and $AF' = (AR', AT')$, such that AF' is a normal expansion of AF, it holds true for the conclusions $A \subseteq AR$ from AF and $A'' \subseteq AR''$ from AF'' that $A \subseteq A''$, where $AF'' = (AR', AT' \setminus \{(a,b) | a \in AR' \setminus AR, b \in A\})$.

Cautious monotony is a well-known property for knowledge-based systems, originally introduced by Gabbay as a generic principle of non-monotonic reasoning [10]. In the context of abstract argumentation, the cautious monotony property stipulates (roughly) that given an AF and a normal expansion of the AF, an argument of the initial AF's conclusion can only be "discarded" if discarding the argument is "caused" by newly added arguments that attack the conclusion. Reference independence ensures that given an AF and a normal expansion of the AF, the conclusion derived from the AF's normal expansion may only "discard" arguments of the original conclusion if newly added arguments are part of the new conclusion; the property is analogous to the *reference independence* property that is defined in the context of micro-economic decision-theory [11, p. 7 et sqq.]. Colloquially speaking, reference independence and cautious monotony can be considered useful to ensure consistency of the conclusions that are generated in an argumentation dialogue: given we infer a conclusion from a framework AF and normally expand the framework, the next conclusion we infer should be aligned with the previous conclusion.

If a semantics does not satisfy reference independence or cautious monotony in itself, DiArg can ensure these properties, for example by adding additional *annihilator arguments* to a framework to reach a conclusion that enables the satisfaction of the principles. Alternatively, DiArg can automatically remove arguments from the framework in a way that ensures principle satisfaction and alters the framework as little as possible.

Let us provide an example to show how reference independence and cautious monotony can be ensured. We start with a sequence that only contains the argumentation framework $AF = (AR, AT) = (\{a,b\}, \{(a,b),(b,a)\})$. The stage extensions of this framework are $\{\{a\},\{b\}\}$. Let us assume that we select the extensions $\{a\}$ as our conclusion. For example, the system can make a random selection if no additional knowledge that can inform the decision exists, or a human user can select the extension based on knowledge that is not modeled within the system; both modes are supported by DiArg. For the *reference independence* scenario, let us assume we add the following normal expansion of AF to the sequence: $AF' = (AR', AT') = (\{a,b,c\}, \{(a,b),(b,a),(b,c),(c,a)\})$. The only stage extension of this framework is $\{b\}$. This violates reference independence, *i.e.*,

[4] See ongoing work for the analogous argumentation principles [5]. In this dialogue system application scenario, we adjust the principle to the fact that we "pick" a particular extension of an argumentation framework as our *conclusion*.



(a) AF. (b) AF'. (c) AF''.

Figure 1. Example argumentation frameworks: $AF = (\{a,b\},\{(a,b),(b,a)\})$, $AF' = (\{a,b,c\},\{(a,b),(b,a),(b,c),(c,a)\})$, and $AF'' = (\{a,b,c\},\{(a,b),(b,a),(b,c),(c,c)\})$. In AF, $\{a\}$ has been accepted as the conclusion. In AF', either $\{a\}$ or $\{c\}$ can be accepted as conflict-free sets that satisfy the reference independence property, given the resolution of AF. In AF'', $\{a\}$ is the only conflict-free set that can satisfy the cautious monotony property, given the resolution of AF.

we discard a from the conclusion and include the originally existing argument b (that was not part of the initial conclusion) without considering the only newly added argument c a part of the new conclusion[5].

For the *cautious monotony* scenario, let us assume we add the following normal expansion of AF to the sequence: $AF'' = (AR'', AT'') = (\{a,b,c\},\{(a,b),(b,a),(b,c),(c,c)\})$. The only stage extension of this framework is $\{b\}$. However, "switching" from conclusion $\{a\}$ to conclusion $\{b\}$ although $\{a\}$ is not attacked by any newly added argument violates cautious monotony. Figure 1 depicts the example's argumentation frameworks.

In both scenarios, we can use the following approaches to ensure reference independence or cautious monotony, respectively:

- Expansionist approach: attack b with an annihilator argument. The annihilator argument is a mere helper, *i.e.*, it is not considered a part of the conclusion.
- Reductionist approach: remove b.

The annihilator approach has the advantage that it can potentially guide the search for new knowledge that allows for a reference independent/cautiously monotonic resolution.

Let us note that the identification of arguments that should either be attacked by the annihilator argument or be removed is relatively straight-forward: we can start by removing (or adding annihilator attacks to) as few arguments in our framework as possible and subsequently increase the number of removed arguments (or annihilator attacks) in case no better solution can be found, starting with the optimistic assumption that in most scenarios, only few arguments will be the "cause" of reference independence or cautious monotony violation. Exploiting specific properties of argumentation semantics' to allow for a more efficient search approach is a promising future research direction.

[5]We assume arguments are either "in" or "out", which is aligned with the *clear preference* property of economically rational decision-making.

3. The DiArg Reasoner

The DiArg reasoner is an open-source Java library, whose source code, including documentation, tests and a tutorial, is available at `https://github.com/Interactive-Intelligent-Systems/diarg`. Let us note that while the specification and analysis of the exact formal foundations of the DiArg reasoner are beyond the scope of this paper, the program code and its documentation allow for an inspection of the underlying data structures and algorithms.

3.1. Abstractions

DiArg provides the following abstractions to manage sequences of argumentation frameworks, the relation between different argumentation frameworks, and their conclusions.

AF tuple. The AF tuple object allows to check whether two AFs are expansions, normal expansions, or (normal) submodules of each other. In addition, the object can, given two argumentation frameworks AF_0 and AF_1, and a conclusion derived from AF_0 with a specific argumentation semantics, determine the largest normal submodules or smallest normal expansions of AF_1, from which a reference independent or cautiously monotonous conclusion w.r.t. AF_0 can be derived.

AF sequence. The AF sequence object allows for the instantiation of AF sequences whose argumentation frameworks satisfy specific properties (*i.e.*, are expanding or normally expanding), and to derive conclusions from any AFs in a sequence, such that the conclusions ensure reference independence or cautious monotony w.r.t. preceding argumentation frameworks and their conclusions.

Context. If context support is activated, the requirement that the sequence must be expanding or normally expanding can be relaxed by specifying that specific sets of arguments are inactive in a specific context. Contexts can be assigned to an AF and managed by a *business logic layer* that implements application-specific program code on top of DiArg. When determining an AF's conclusion that satisfies reference independence or cautious monotony, DiArg searches for the AF's most recent predecessor whose contexts are consistent with the contexts of the AF.

Serializer. The serializer supports the export of argumentation frameworks, sequences, and extensions in a JavaScript Object Notation (JSON)-based format, as well as the instantiation of the corresponding DiArg objects from JSON.

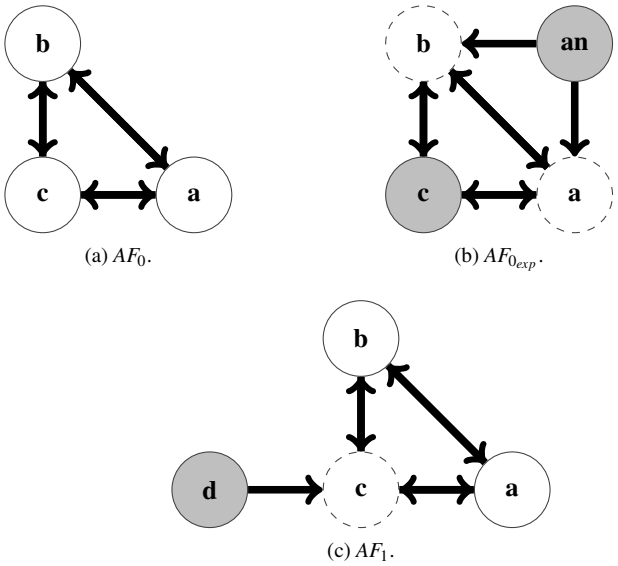
3.2. Demonstration Example

To show how DiArg can be applied, let us introduce the following example. Let us assume we are developing a digital assistant for stress management (see Guerrero *et al.* [12] for a related application scenario). The assistant recommends stress-relieving activities (represented by arguments) to an end-user; the end-user can then either accept the recommendation and add it to their schedule or reject the activity by adding an argument that attacks the corresponding activity. This argument will be considered when providing future recommendations. The argument may be context-dependent. For example, a user may reject stress-relieving activities that are particularly time-consuming during weekdays, but accept them at weekends.

Let us present a simplified example of how DiArg can generate recommendations for a specific sub-scenario. We start with an AF that has three arguments representing the execution of different activities. Because at any point in time, only one activity can be executed, all arguments attack each other; *i.e.*, we have the argumentation framework $AF_0 = (\{a,b,c\},\{(a,b),(b,a),(b,c),(c,b),(c,a),(a,c)\})$, where we can interpret a, b, and c as follows: a: recommend activity *meditate*; b: recommend activity *join social lunch*; c: recommend activity *go hiking*. We assume we have configured the sequence object to have the following properties: *i)* We use stage semantics σ_{stage}. *ii)* In the sequence, each AF must be a normal expansion of its predecessor. *iii)* The conclusion of an AF is determined by expanding the AF until a unique (exactly one) extension is determined that implies reference independence w.r.t. the previous conclusion.

The recommender system can either allow the user to choose any of the extensions a specific semantics returns for the AF, or it can add an *annihilator argument* to the AF that (with its attacks on other arguments) enforces a unique extension. This behavior is specified on top of the sequence (*i.e.*, DiArg supports both variants and the specific implementation depends on how the DiArg library is integrated into the application). Let us assume our implementation does the latter. To resolve the initial AF, the system adds an annihilator argument *an* to generate a normal expansion for which stage semantics returns exactly one extension E' and $\exists E \in \sigma_{stage}(AF)$, such that $E = E' \setminus \{an\}$. In our case, the system adds the argument *an* to the AF, and the attacks (an,b) and (an,a), *i.e.*, $AF_{0_{exp}} = (\{a,b,c,an\},\{(a,b),(b,a),(b,c),(c,b),(c,a),(a,c),(an,a),(an,b)\})$. It follows that the conclusion is $\{an,c\}$ ($\sigma_{stage}(AF_{0_{exp}}) = \{\{an,c\}\}$), *i.e.*, the conclusion is $\{c\}$ when excluding the annihilator argument *an*.

In the example scenario, the conclusion $\{c\}$ implies that our application suggests *Go hiking* as the stress-relieving activity to the end-user. Let us assume the user wants to reject the recommendation because she does not have time to go hiking on weekdays. For this, she inserts this feedback through the system's user interface, which generates the next AF in our sequence: $AF_1 = (\{a,b,c,d\},\{(a,b),(b,a),(b,c),(c,b),(c,a),(a,c),(d,c)\})$. On the business logic layer, the developer can specify that the provided user feedback will be inactive when a new recommendation is generated for weekend activities. However, in the current context, the system resolves AF_1. Because of the attack cycle "a attacks b attacks a", the system again uses an annihilator argument to generate a single recommendation (exactly one extension), which can be either $\{a\}$ or $\{b\}$. When providing a "weekend" recommendation, the AF is resolved in a consistent manner w.r.t. to its closest predecessor with whose contexts it is aligned; *i.e.*, if we add a new framework $AF_2 := AF_1$ and activate the "weekend" context, the system again generates the initial recommendation $\{c\}$; *i.e.*, argument d and its attack relations are ignored and the conclusion $\{c\}$ is consistent with (in this case: identical to) the initial conclusion inferred from AF_0. Figure 2 depicts the example's argumentation frameworks.



(a) AF_0. (b) $AF_{0_{exp}}$. (c) AF_1.

Figure 2. Recommender system scenario: Given $AF_0 = (\{a,b,c\},\{(a,b),(b,a),(b,c),(c,b),(c,a),(a,c)\})$, the system auto-generates the annihilator argument *an* and the attacks (an,a) and (an,b) to "select" the extension $\{c\} \in \sigma_{stage}(AF_0)$ ($\{c,an\} \in \sigma_{stage}(AF_{0_{exp}})$). Then, a human user creates AF_1 by adding the argument d and the attack (d,c) to AF_0 to indicate that the recommendation $\{c\}$ is not useful in the current context.

4. Limitations and Future Work

In this paper, we have presented ongoing work on an argumentation-based dialogue reasoner that iteratively resolves sequences of abstract argumentation frameworks. The following enhancements can be considered useful future work:

Improved context support. DiArg uses *context* as a means to provide an additional tool for allowing *business logic* (application-specific program code) to relax constraints on how specific AFs in a sequence are resolved; *i.e.*, in DiArg, abstract argumentation can be considered the lowest layer of abstraction. This approach stands in contrast with many formal argumentation methods that are concerned with the internal structure of arguments, but is well-aligned with methods from other domains, *e.g.*, with the mapping between business process diagrams to Petri nets as the lowest level abstraction layer [13]. Further investigating the integration of formal argumentation methods with application-specific program code layers and their paradigms can be promising future work.

Integration with recommender systems approaches. A relevant use case of DiArg may be the enhancement of argumentation-based recommender systems, which are often implemented using Machine Learning (ML)-based methods. For example, DiArg can potentially help address "cold start" issues and facilitate the incorporation of user feedback. Future work can put DiArg into the context of ML-based or hybrid approaches like the argumentation-based recommender system introduced by Rago *et al.* [1].

Interoperability enhancements in alignment with the argument interchange format. DiArg's serializer supports the import and export of argumentation frameworks and extensions, and of DiArg-specific objects like context and argumentation sequences. So far, the serializer does not consider the standardization attempts that have been made

in the form of the Argument Interchange Format (AIF) [14]. A proposal of how to serialize argumentation frameworks as JSON objects exists in the context of AIF[6]. Consequently, an assessment to what extent DiArg can comply with the standardization approaches of AIF can be considered relevant future work.

References

[1] A. Rago, O. Cocarascu and F. Toni, Argumentation-Based Recommendations: Fantastic Explanations and How to Find Them, in: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, AAAI Press, 2018, pp. 1949–1955–. ISBN ISBN 9780999241127.

[2] M. Thimm, Tweety: A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation, in: *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'14, AAAI Press, 2014, pp. 528–537–. ISBN ISBN 1577356578.

[3] T. Rienstra, M. Thimm and N. Oren, Opponent Models with Uncertainty for Strategic Argumentation, in: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, AAAI Press, 2013, pp. 332–338–. ISBN ISBN 9781577356332.

[4] E. Awad, M.W.A. Caminada, G. Pigozzi, M. Podlaszewski and I. Rahwan, Pareto optimality and strategy-proofness in group argument evaluation, *Journal of Logic and Computation* **27**(8) (2017), 2581–2609. doi:10.1093/logcom/exx017.

[5] T. Kampik and J.C. Nieves, Abstract Argumentation and the Rational Man, 2019.

[6] T. Kampik, Economic Rationality and Abstract Argumentation, in: *Online Handbook of Argumentation for AI: Volume 1*, F. Castagna, F. Mosca, J. Mumford, S. Sarkadi and A. Xydis, eds, arXiv, 2020, pp. 7–11, eprint at https://arxiv.org/abs/2006.12020.

[7] P.M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, *Artificial Intelligence* **77**(2) (1995), 321–357. doi:https://doi.org/10.1016/0004-3702(94)00041-X. `http://www.sciencedirect.com/science/article/pii/000437029400041X`.

[8] B. Verheij, Two Approaches to Dialectical Argumentation: Admissible Sets and Argumentation Stages, in: *In Proceedings of the biannual International Conference on Formal and Applied Practical Reasoning (FAPR) workshop*, Universiteit, 1996, pp. 357–368.

[9] R. Baumann and G. Brewka, Expanding Argumentation Frameworks: Enforcing and Monotonicity Results., *COMMA* **10** (2010), 75–86.

[10] D.M. Gabbay, Theoretical Foundations for Non-Monotonic Reasoning in Expert Systems, in: *Logics and Models of Concurrent Systems*, K.R. Apt, ed., Springer Berlin Heidelberg, Berlin, Heidelberg, 1985, pp. 439–457. ISBN ISBN 978-3-642-82453-1.

[11] A. Rubinstein, *Modeling bounded rationality*, MIT press, 1998.

[12] E. Guerrero, J.C. Nieves and H. Lindgren, An activity-centric argumentation framework for assistive technology aimed at improving health, *Argument & Computation* **7**(1) (2016), 5–33.

[13] R.M. Dijkman, M. Dumas and C. Ouyang, Semantics and analysis of business process models in BPMN, *Information and Software Technology* **50**(12) (2008), 1281–1294. doi:https://doi.org/10.1016/j.infsof.2008.02.006. `http://www.sciencedirect.com/science/article/pii/S0950584908000323`.

[14] I. Rahwan and C. Reed, *The Argument Interchange Format*, in: *Argumentation in Artificial Intelligence*, G. Simari and I. Rahwan, eds, Springer US, Boston, MA, 2009, pp. 383–402. ISBN ISBN 978-0-387-98197-0. `https://doi.org/10.1007/978-0-387-98197-0_19`.

[6] See `https://arg-tech.org/index.php/projects/aifbdb-user-guide/`