KI - Künstliche Intelligenz (2018) 32:209–212
https://doi.org/10.1007/s13218-018-0532-1

PROJECT REPORT

ArgueApply: Abstract Argumentation at Your Fingertips

Jörg Pührer[1]

Published online: 15 May 2018

Abstract
We report on `ArgueApply`, a mobile app for argumentation based on the Grappa framework. With `ArgueApply` users can engage in online discussions and evaluate their semantics. Being a mobile tool, it is intended to be more accessible than existing systems for computing argumentation semantics allowing, e.g., for spontaneous analysis of an ongoing discussion or collective preparation for an important debate. `ArgueApply` uses answer-set programming for locally evaluating the semantics on the mobile device.

Keywords Argumentation · Grappa · Answer-set programming · Mobile app · Android

1 Introduction

Argumentation is an area at the intersection of Philosophy and many subfields of Artificial Intelligence (AI) such as knowledge representation, nonmonotonic reasoning, and multi-agent systems and deals with the question of how conclusions can be reached through logical reasoning [1]. The field has seen a steady rise of interest over the last two decades which paved the way for computational tools for argumentation. In this work, we present `ArgueApply`, a mobile application that makes use of recent advancements in formal argumentation. It allows users to participate in online debates and to analyse these discussions as well as their inherent viewpoints revealed by argumentation semantics. To this end, `ArgueApply` uses the Grappa framework [4], a recent extension of argumentation frameworks (AFs) by Dung [5]. AFs represent argumentation scenarios as simple directed graphs, where each node represents an argument and edges between arguments indicate how they attack each other. Despite the simplicity of AFs, different interesting semantics can be defined for this formalism. Yet, AFs come with limited expressive capabilities as they only allow for one type of relation between arguments, the binary attack relation. Grappa was introduced to make the powerful argumentation semantics of abstract dialectical frameworks [3] more accessible to users. Similar to AFs, Grappa frameworks are based on directed graphs but edges carry arbitrary labels that allow for discriminating different types of relations between nodes (a Grappa node stands for a statement in a discussion and corresponds to an argument in an AF). Moreover, Grappa offers a pattern language for customising statement acceptance: every node has an acceptance pattern assigned that determines whether the statement is accepted or rejected based on the labels of its incoming edges. For example a statement with the pattern $\#(+) - \#(-) > 0$ is accepted if there are more incoming edges from accepted statements that are labelled with + than edges labelled −.

With `ArgueApply` we want to go one step further in usability and offer argumentation technology for the lay user. It allows users to participate in online debates and to analyse these discussions as well as their inherent viewpoints revealed by argumentation semantics.

The evaluation of the semantics in `ArgueApply` is based on answer-set programming (ASP) [8–10]. We use transformations from Grappa to ASP that exploit the aggregation capabilities of ASP [2]. As a solving backend we rely on `clingo` [7]. The solver is executed locally on the mobile device and results are stored in an online cache such that fellow users do not need to recompute. `ArgueApply` is available for the Android platform and written primarily in Java. An overview of the technology used in `ArgueApply` is given in Fig. 1. We implemented an ASP service that may run independently of the Android application and encapsulates a JavaScript version of `clingo` 5.2. The availability of a JavaScript interpreter by default on the Android

✉ Jörg Pührer
jpuehrer@dbai.tuwien.ac.at

[1] Institute of Logic and Computation, TU Wien, Favoritenstraße 9, Vienna, Austria

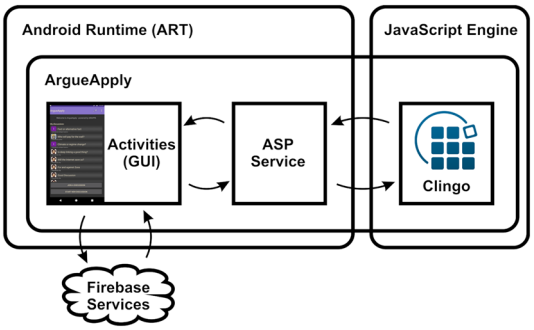

Fig. 1 Technology overview for `ArgueApply`

platform guarantees that we can use the solver without the need to compile it natively on every device. `ArgueApply` builds on the web application platform Firebase [6] which provides user authentication and a non-relational database storing all online data.

2 Functionality and User Interface

Users of `ArgueApply` can participate in online discussions, each of which implicitly corresponds to a Grappa instance. As our app focuses on lay users, no formal background is required to use it. Every participant can post new statements in a similar way as sending messages in a chat application.

Participants of the discussion can express which influence they see between two statements by chosing one of the link labels, ++,+, −, and − that express strong support, support, attacks, and strong attacks, respectively. The active label is then chosen by plurality voting, followed by weighting in case of multiple winning labels.

A discussion is displayed as an expandable list of its statements as depicted in Fig. 2. When a statement item is expanded, the links of the statement are shown, by listing its influencing statements and the respective edge labels.

We implemented four Grappa semantics [4] using a translational approach that was introduced in a recent paper [2]: admissible, complete, grounded and preferred semantics but only the latter is used under default settings.

`ArgueApply` uses two closely related acceptance patterns:

$$2 * \#(++) + \#(+) - 2 * \#(--) - \#(-) \geq 0$$

and

$$2 * \#(++) + \#(+) - 2 * \#(--) - \#(-) > 0.$$

The patterns express that the statement should be accepted if there are more positive links than negative links from acceptance statements, where labels with doubled symbols

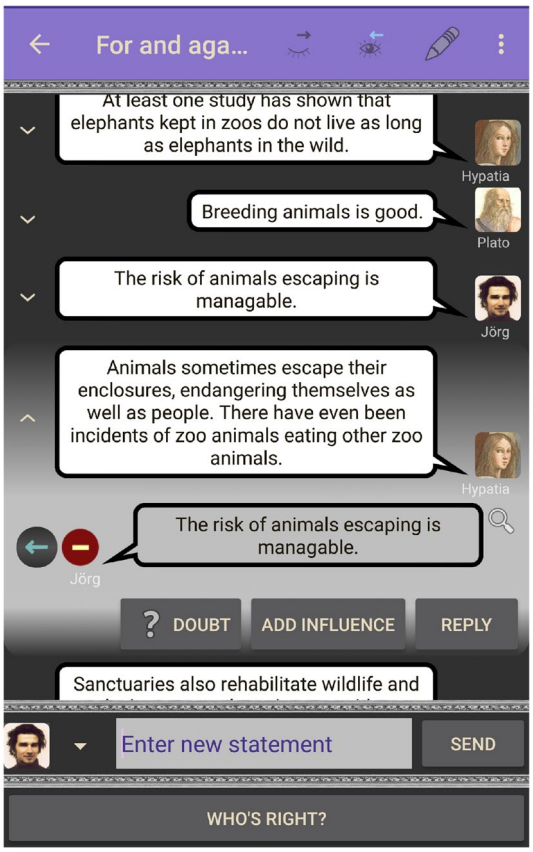

Fig. 2 A discussion in `ArgueApply`. All statements are shown in an expandable list. On expansion of a statement item its links are displayed. Here, the fourth statement is expanded, showing one incoming edge labelled −

count twice as positive, respectively negative, than their single counterparts. The first pattern accepts statements with a neutral balance. By default, we use this pattern for all arguments. Our rationale is that a new statement in a debate that is not known to be influenced by any other statement, i.e., being unquestioned, should be accepted. Sometimes there are statements for which we do not know whether they are true but we do not have any appropriate counterargument. For this case, a user can express general doubt in a statement. If one of the participants of the discussion doubts a statement, two modifications happen:

1. the statement is evaluated with the second acceptance pattern and
2. a link labelled + is added from the statement to itself.

As a consequence, when no further link ends at the statement s, it is not by default accepted, but leads to resulting interpretations where it is accepted and others where it is rejected under the semantics we implemented (except for grounded semantics), similar like a choice atom $\{s\}$ in

answer-set programming. The self-link is not shown as a further link to the user (as this might be confusing), however users get some indication whether someone doubted the statement.

Example 1 Consider an `ArgueApply` discussion with two statements:

- a : There is no wall.
- b : Atoms are invisible, the wall is made of atoms, therefore the wall is invisible.

Without any vote for a link, there is one solution (under preferred semantics) where both a and b are accepted. When a majority votes for a link labelled − from b to a, then we have a solution accepting b and rejecting a. When some participant expresses doubt for statement b, we get an additional solution where a is accepted and b is rejected. Finally, if someone adds a further statement

- c : I can see the wall.

attacking both a and b, we drop again to a single solution that accepts c and rejects the other statements.

A user can trigger the evaluation of the Grappa instance at all times and then browse through the resulting interpretations such that for each statement it is highlighted whether it is accepted or rejected with respected to the currently selected solution (as in Fig. 3).

We see different usage scenarios for `ArgueApply`. For instance, if a group has a dispute and a neutral point of view is required, each person may enter her contributing statements in an `ArgueApply` discussion, and collectively they determine the link structure. The evaluation reveals sets of arguments that are jointly acceptable, indicating who has a point of view on the debate that is logically consistent (wrt. to Grappa semantics). In this application scenario, `ArgueApply` acts as a neutral instance, e.g., deciding which decision should be taken by the overall group. Here, `ArgueApply` offers the advantage over simple voting for a decision that the group acts on the basis of clearly structured arguments rather than the average of intuitions. Thus it makes the motivation for group decisions more transparent, documentable, and verifiable.

Another scenario is preparation for a debate, e.g., when a representative of a group is invited to a public debate. In the run-up to the event, members of the group use `ArgueApply` to find out an optimal line of argumentation as preparation for the representative. In a brainstorming process, arguments for and against the cause of the group are collected as statements of an `ArgueApply` discussion. For this type of application, `ArgueApply` allows for using pre-defined avatars for posting statements that do not reflect one's own opinion (see Fig. 4). If the evaluation reveals that there is a line of argumentation of the opponent that is consistent, the team can search for new arguments that break this line.

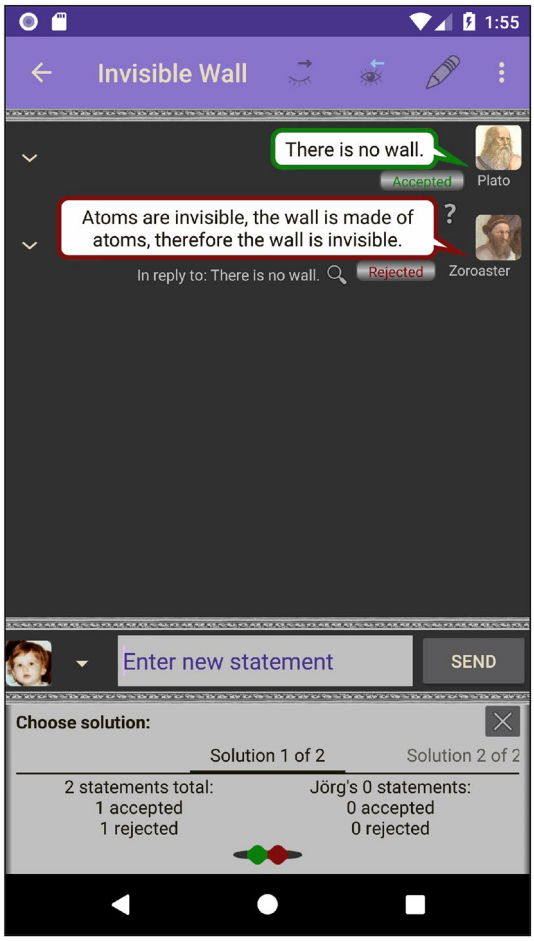

Fig. 3 After evaluating the Grappa instance, the user can select which resulting interpretation to view in the bottom area. The statements in the list are coloured and annotated according to their acceptance under the chosen solution. The Grappa instance is that from Example 1 before adding the third statement

For a discussion on further application scenarios we refer the interested reader to a previous paper [11] on `ArgueApply`.

3 Conclusion and Outlook

To the best of our knowledge, `ArgueApply` is the first mobile app that allows users to collaboratively formalise debates and evaluate them based on techniques from formal argumentation theory and it is publicly available.[1] In future

[1] https://www.informatik.uni-leipzig.de/~puehrer/ArgueApply/ or Google Play

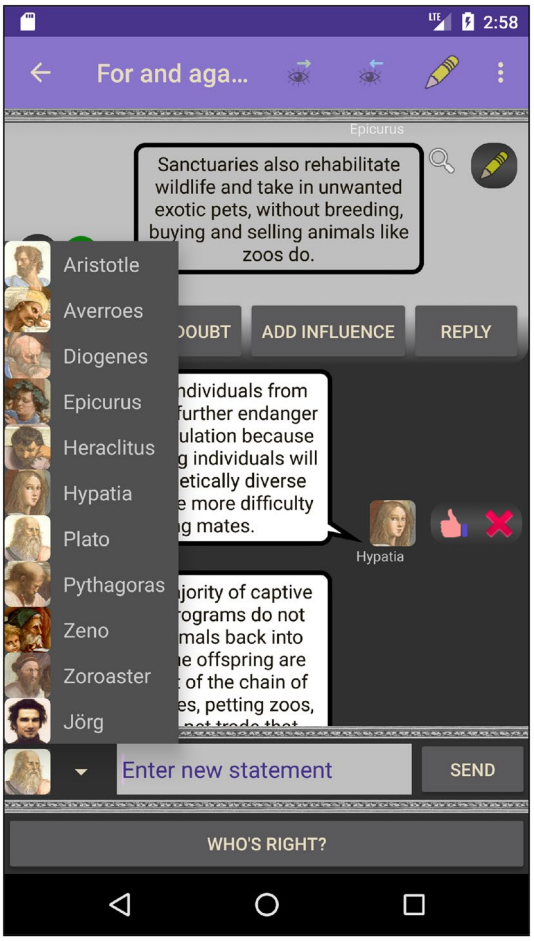

Fig. 4 Besides the user account, ten philosophers can be selected as avatars for publishing statements. These are used for modelling opinions that no human participant of the discussion shares

work, we want to explore enforcing features for Grappa to improve the system's capabilities for scenario like the one for preparing a debate: What type of arguments do I need to add or alter in order to improve the evaluation results for my party?

Acknowledgements This work was supported by the German Research Foundation (DFG) under Grant BR 1817/7-2 and by the Austrian Science Fund (FWF) under Grant I2854.

References

1. Baroni P, Gabbay D, Giacomin M, van der Torre L (2018) (eds) Handbook of formal argumentation. College Publications. http://formalargumentation.org**(to appear)**
2. Brewka G, Diller M, Heissenberger G, Linsbichler T, Woltran S (2017) Solving advanced argumentation problems with answer-set programming. In: Singh S, Markovitch S (eds) Proceedings of AAAI 2017, pp 1077–1083
3. Brewka G, Strass H, Ellmauthaler S, Wallner JP, Woltran S (2013) Abstract dialectical frameworks revisited. In: Proceedings of IJCAI 2013, pp 803–809
4. Brewka G, Woltran S (2014) GRAPPA: a semantical framework for graph-based argument processing. In: Proceedings of ECAI 2014, pp 153–158
5. Dung PM (1995) On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif Intell 77(2):321–358
6. Firebase Platform (2018). https://firebase.google.com/. Accessed 14 Mar 2018
7. Gebser M, Kaminski R, Kaufmann B, Ostrowski M, Schaub T, Schneider M (2011) Potassco: the potsdam answer set solving collection. AI Commun 24(2):107–124
8. Gebser M, Kaminski R, Kaufmann B, Schaub T (2012) Answer set solving in practice. Synthesis lectures on artificial intelligence and machine learning. Morgan and Claypool, San Rafael
9. Marek VW, Truszczyński M (1999) Stable models and an alternative logic programming paradigm. In the logic programming paradigm: a 25-year perspective. Springer, Berlin, pp 375–398
10. Niemelä I (1999) Logic programs with stable model semantics as a constraint programming paradigm. Ann Math Artif Intell 25(3–4):241–273
11. Pührer J (2017) ArgueApply: a mobile app for argumentation. In: Balduccini M, Janhunen T (eds) Proceedings of LPNMR 2017, LNCS, vol 10377. Springer, Berlin, pp 250–262