Computational Models of Argument
F. Toni et al. (Eds.)

doi:10.3233/FAIA220178

Gorgias Cloud: On-line Explainable Argumentation

Nikolaos I. SPANOUDAKIS [a,1], Georgios GLIGORIS [a] Antonis C. KAKAS [b] and Adamos KOUMI [b]
[a] *Technical University of Crete, Greece*
[b] *University of Cyprus, Cyprus*

Keywords. Computational Argumentation, SaaS, IDE, xAI

Gorgias Cloud offers argumentation-based decision making as a service. The service includes an integrated development environment for the theories, testing and execution based on user scenarios, and, finally, an API for use by user applications.

Gorgias is a structured argumentation framework where arguments are constructed using a basic scheme of *argument rules*. Two types of arguments are constructed within a Gorgias argumentation theory: *object-level* arguments and *priority arguments* expressing a preference between other arguments. Admissible *composite arguments* supporting a claim typically include both types of arguments. The Gorgias framework was introduced in [1], extended in [2] and applied to a variety of real-life application problems in [3].

The Gorgias system allows us to code argumentation theories of the form described above and subsequently query the system to find out if there is an admissible (composite) argument that supports a desired Claim. The system of Gorgias has been publicly available since 2003 and has been used by several research groups to develop prototype real-life applications of argumentation in a variety of application domains. Today, it is available as a service over the internet with Gorgias Cloud, which provides an integrated environment for developing applications of argumentation with three novel features:

1. Assistance for editing argumentation theories in the internal code language of Gorgias using templates for object level or priority arguments and abducibles
2. Ability to store multiple scenarios to test the behaviour of developed theories
3. REST-compliant web API so that Gorgias queries can be executed from any other programming environments (e.g. Java, Python) used for developing applications.

The code shown on the left hand side of the Gorgias Cloud Execution Panel in Figure 1, shows a simple example of an argumentation theory. Gorgias rules are in the form $rule(label, conclusion, supporting\ information)$. Those with labels $r1(X)$ and $r2(X)$ are for and against buying an object. The priority argument rules $pr1(X), pr2(X)$ support one or the other of the object-level rules, depending on whether we are low on funds.

Let us assume that we have an object, obj, for which $need(obj)$ and $neg(urgent\_need(obj))$ hold. The query for asking if there is an admissible composite argument supporting the conclusion to not buy the obj, i.e. $neg(buy(obj))$, is posed at

[1]Corresponding Author: Nikolaos I. Spanoudakis, Applied Mathematics and Computers Laboratory, Technical University of Crete, University Campus, 73100 Chania, Greece; E-mail: nikos@amcl.tuc.gr.

Figure 1. Execution Panel of Cloud Gorgias, the theory on the left and the results of the query in the center.

the prompt at the bottom of the Execution Panel (see Figure 1). In the Execution Panel we see both a) the *Internal Explanation* (IE) of the composite argument E, $Explanation = [ass(lowOnFunds), f1, pr1(obj), r2(obj)]$, where $f1$ is the label of the belief $neg(urgent\_need(obj))$, as well as b) the *Application Level Explanation* (ALE) in a human-friendly format. ALE starts by presenting the Claim, followed by the supporting information of the object level rule that supports it. Then, if this Claim has been strengthened by a preference rule over another position, then this conflicting position is printed along with the supporting information of the preference rule. Finally, the user is informed that any employed assumptions should be confirmed.

The generation of the ALE from the IE is an important feature of the Gorgias Cloud as it exhibits the desired characteristics of being *attributive, contrastive and actionable*:

- **Attributive:** Extracted from the object-level argument rules in E.
- **Contrastive:** Extracted from the priority argument rules in E.
- **Actionable:** Extracted from the hypothetical or abducible arguments in E.

Developers and users can use the ALEs to evaluate the behaviour of their system with respect to its specifications. This is made easy as these explanations are at the same high cognitive and language level as that of the application domain of the system.

References

[1] Kakas AC, Mancarella P, Dung PM. The Acceptability Semantics for Logic Programs. In: The 11th International Joint Conference on Logic Programming; 1994. p. 504-19.

[2] Kakas AC, Moraitis P. Argumentation based decision making for autonomous agents. In: The Second International Joint Conference on Autonomous Agents & Multiagent Systems, (AAMAS 2003), July 14-18, Melbourne, Victoria, Australia. ACM; 2003. p. 883-90.

[3] Kakas AC, Moraitis P, Spanoudakis NI. *GORGIAS*: Applying argumentation. Argument & Computation. 2019;10(1):55-81.