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*

The Gorgias Cloud service 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 user applications.

Gorgias is a structured argumentation framework where arguments are constructed using a basic (content independent) scheme of *argument rules*. Two types of arguments are constructed within a Gorgias argumentation theory: *object-level* arguments and *priority arguments* expressing a preference, or relative strength, between other arguments. The dialectic argumentation process of Gorgias to determine the acceptability/admissibility of an argument supporting a desirable claim typically occurs between *composite arguments* where priority arguments are included into the composite argument in order to strengthen the arguments currently committed to. The Gorgias framework was introduced in [1], extended in [2] and applied to a variety of real-life application problems [3].

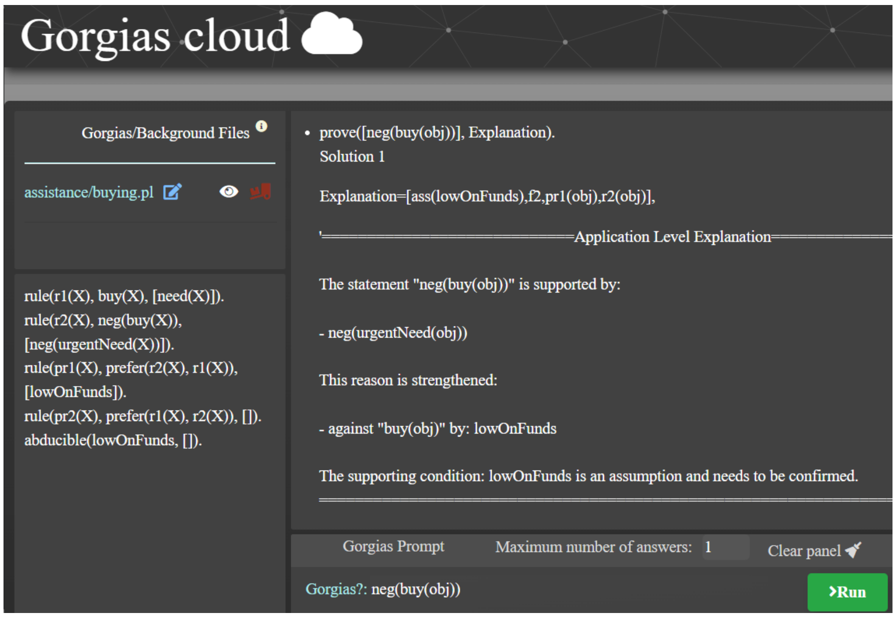
The Gorgias system allows us to code argumentation theories of the form described above and subsequently query the system to find out if there is an admissible (composite) argument that supports a desired Claim. The system of Gorgias has been publicly available since 2003 and has been used by several 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 the 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 sup-

[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.



```
rule(r1(X), buy(X), [need(X)]).
rule(r2(X), neg(buy(X)),
[neg(urgentNeed(X))]).
rule(pr1(X), prefer(r2(X), r1(X)),
[lowOnFunds]).
rule(pr2(X), prefer(r1(X), r2(X)), []).
abducible(lowOnFunds, []).
```

```
• prove([neg(buy(obj))], Explanation).
  Solution 1

  Explanation=[ass(lowOnFunds),f2,pr1(obj),r2(obj)],

  '==========================Application Level Explanation==========

  The statement "neg(buy(obj))" is supported by:

  - neg(urgentNeed(obj))

  This reason is strengthened:

  - against "buy(obj)" by: lowOnFunds

  The supporting condition: lowOnFunds is an assumption and needs to be confirmed.
  ==================================================================
```

```
Gorgias?: neg(buy(obj))
```

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

porting the conclusion to not buy the obj, i.e. $neg(buy(obj))$, is posed at the prompt at the bottom of the Execution Panel (see Figure 1). In the center of the Execution Panel we can see both an *Internal Explanation* of the composite argument E, i.e. $Explanation = [ass(lowOnFunds), f1, pr1(obj), r2(obj)]$, as well as an *Application Level Explanation* in a human-friendly format. This is an important feature of the Gorgias Cloud system as the internal, machine friendly, explanation of a composite argument is naturally translated into an application level explanation exhibiting 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 application level explanations to evaluate the behaviour of their system with respect to the specification requirements under which the system is developed. 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] Antonis C. Kakas, Paolo Mancarella, and Phan Minh Dung. The acceptability semantics for logic programs. In *Proc. of 11th Int. Conf. on Logic Programming*, pages 504–519, 1994.

[2] Antonis C. Kakas and Pavlos Moraitis. Argumentation based decision making for autonomous agents. In *The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003, July 14-18, 2003, Melbourne, Victoria, Australia, Proceedings*, pages 883–890. ACM, 2003.

[3] Antonis C. Kakas, Pavlos Moraitis, and Nikolaos I. Spanoudakis. *GORGIAS*: Applying argumentation. *Argument Comput.*, 10(1):55–81, 2019.