Application Level Explanations for Argumentation-based Decision Making

Nikolaos I. Spanoudakis[1], Antonis C. Kakas[2] and Adamos Koumi[2]

[1]*School of Production Engineering and Management, Technical University of Crete, University Campus, Chania, 73100, Greece*
[2]*Department of Computer Science, University of Cyprus, 75 Kallipoleos Str., P.O. Box 537, Nicosia, CY-1678, Cyprus*

Abstract
In this paper we explore the explainability of the Gorgias argumentation framework. Gorgias is being used for automated decision-making or decision-support in real-world systems development. The dialectical argumentation reasoning within the Gorgias framework gives, besides an admissibly supported Position, an internal representation of the argumentative reasoning that leads to this. This representation, can be manipulated by applications to produce case-based human readable explanations. It is also employed by the Gorgias Cloud web-based integrated application development environment that facilitates the development of argumentation-based systems over the internet for providing human-readable explanations. These explanations can assist both in the development and in the validation of the theory capturing the knowledge of an application.

Keywords
Explainable AI, Argumentation-based Decision Making, Software as a Service, Decision Support Systems

1. Introduction

Explainable Artificial Intelligence (xAI) has become a major trend with numerous approaches for generating useful explanations of decisions taken by AI systems. The field concerns both the task of explaining the predictions of black box machine learning systems [1] or, more generally, the task of providing the key reasons for the decision of a system [2]. Systems need not only to perform well in the accuracy of their output but also in the interpretability and understandability of their results. Explanations of systems facilitate their usability by other systems (artificial or human) and contribute significantly towards building a high level of trust towards information systems. They have a role to play both at the level of developing a system and at the level of acceptance of the system in its application environment.

In comparison with explanations in Expert Systems of the early era of AI, the emphasis now is on explanations that are cognitively compatible with the users (and developers) of systems, providing useful information for the further deployment of the system's decisions. The "human in the loop" paradigm of modern AI means that humans want, need and are entitled to

1st International Workshop on Argumentation for eXplainable AI (ArgXAI, co-located with COMMA '22), September 12, 2022, Cardiff, UK
nispanoudakis@tuc.gr (N. I. Spanoudakis); antonis@ucy.ac.cy (A. C. Kakas); akoumi01@cs.ucy.ac.cy (A. Koumi)
https://users.isc.tuc.gr/~nispanoudakis (N. I. Spanoudakis); http://www.cs.ucy.ac.cy/~antonis/ (A. C. Kakas)
0000-0002-4957-9194 (N. I. Spanoudakis)

explanations, particularly when decisions can affect them significantly. Explanations are, thus, aimed at rendering our systems transparent, accountable and contestable. According to the General Data Protection Regulation (GDPR) of the European Parliament [3], a human should be able to "contest such decisions decision". Without an explanation at the cognitive level of the user this right to contest cannot be exercised in any meaningful way.

Automated decision-making based on computational argumentation can benefit from the latter's natural approach to explaining the reasons behind the acceptance of an argument. The result of argumentation is typically some coalition of arguments that together support acceptably a desired conclusion or position. Irrespective of the particular notion of acceptability that we adopt, these coalitions, or cases of support, can be unraveled to produce an explanation that contains information both at the level of the basic support of the position, and at the level of the relative strength of the position in contrast to other possible alternative positions. Recent surveys on Explainable Argumentation [4, 5] analyze this link between argumentation and explanation and explore its potential.

In this paper we study how argumentation-based explanations can be systematically constructed and adapted to the different needs of real-life applications within the structured argumentation framework of Gorgias [6]. We consider the problem both at the general level, i.e. independently from the particular application domain, but also at the local level where the specific needs and features of a particular domain need to be considered. Subsequently, we present real-world applications showing how they can utilize the dialectic nature of argumentation-based reasoning to come up with explanations aimed at assisting human decision makers, e.g. by explaining a system's position as a peer expert assistant, for the human expert to take a final, more informed, decision.

Medica [7], for example, is an argumentation system built using the Gorgias framework, that allows for deciding if a specific person can have access to sensitive medical files, based on a) who is the requester, e.g. the owner, a medical doctor, etc, b) what is the reason for requesting access (research, treatment, etc), and, c) what additional support is available, e.g. order from the medical association, written consent from the owner and other similar requests. Argumentation allows such decisions to be *explainable* to humans [7] and assist them by citing the particular parts of the legislation that justifies the system's decision. The system also provides actionable explanations showing what extra information is needed for a desired level of access to be granted. The *Gynecological AI Diagnostic Assistant (GAID)* [8] provides appropriate explanations for its medical diagnosis that aim to assist obstetricians in a medical clinic.

The above systems are based on the Gorgias framework, which was introduced in 1994 [9], extended in 2003 [6] and has since been applied to a variety of real-life application problems [10]. Recently, Gorgias Cloud [11] was developed as a web-based Integrated Development Environment (IDE) for applications of argumentation, offering an argumentation-based reasoning Application Programming Interface (API). This allows us to build systems that use argumentation *as a service* in the context of decision making applications. Moreover, Gorgias Cloud provides an explanation capability that facilitates experts to judge whether a decision theory functions as it should, and application developers to produce human-readable explanations for their users.

In what follows, we outline the Gorgias framework and provide some useful background definitions and notions of explainable AI. Then, we show how Gorgias captures composite (object-level as well as priority) arguments in its internal representation together with a human-readable format, which we call Application-level Explanations. In section 4, we discuss some examples of explanations for real-world application. Section five concludes.

2. Background

2.1. The Gorgias Argumentation Framework

Gorgias is a structured argumentation framework where arguments are constructed using a basic (content independent) argument scheme that associates a set of premises with the claim, or position, of the argument. In this framework we can represent **argument rules**, denoted by **Premises** $\triangleright$ **Position** linking the Premises, a set of literals, to a literal Position: we say that the argument rule **supports** the Position. The Premises are typically given by a set of conditions describing a scenario. Some of these conditions can be defeasible, called **beliefs**, and they can be argued for or against.

An **argument**, A, is a set of **argument rules**. In an argument, A, through the successive application of its constituent argument rules, several claims, including a "final or desired" claim (or position), are reached and, hence, supported by A. Argument rules have the syntax of Extended Logic Programming, i.e. rules whose conditions and conclusion (claim) are positive or explicit negative atomic statements, but where negation as failure is excluded from the language[1]. The conclusion of an argument rule can be a positive or negative atomic statement. The conflict relation in Gorgias can be expressed either through explicit negation, or through a complementarity relation between statements in the application language, or through an explicit statement of conflict between argument rules, or, finally, through any combination of the previous three ways.

Two types of arguments are constructed within a Gorgias argumentation theory: a) **object-level** arguments and b) **priority** arguments, the latter 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. Priority arguments have the same syntactic form with other arguments rules, however, the Position they support is a statement of priority between two other individual argument rules. These arguments rules for which a priority is expressed can themselves be priority argument rules, in which case we say that the priority argument is a **higher-order** priority argument expressing the fact that we prefer one priority over another, in the context of the premises of this higher-order priority.

Consider, for example, the rules in Listing 1. They are given in the form:

$$rule(label, claim, defeasiblePremises){:}\text{-}nonDefeasiblePremises.$$

The **object-level** arguments are those with labels $r1(X)$ and $r2(X)$. Rule $r1(X)$ supports the claim to buy *X*, expressed with the predicate *buy(X)*, with the non-defeasible premise that

[1]Initially, the framework of Gorgias had the name $LPwNF$: Logic Programming without Negation as failure.

Listing 1: The "buying.pl" code.

```
1 rule(r1(X), buy(X), []):-need(X).
2 rule(r2(X), neg(buy(X)), [neg(urgentNeed(X))]).
3 rule(pr1(X), prefer(r2(X), r1(X)), []):-lowOnFunds.
4 rule(pr2(X), prefer(r1(X), r2(X)), []).
5 abducible(neg(urgentNeed(X)), []).
6 abducible(urgentNeed(X), []).
```

the user needs *X*, expressed with the predicate *need(X)*. Respectively, rule $r2(X)$ supports the claim to not buy *X*, expressed with predicate *neg(buy(X))*, with the defeasible premise that the need for X is not urgent, expressed with the predicate *neg(urgentNeed(X))*. Note that defeasible premises could be themselves the claims of other object-level arguments, or can be hypothesised as **abducibles**. For example, the *urgentNeed(X)* and its negation *neg(urgentNeed(X))* are defined as hypothesis in lines 5-6. This approach is usually followed when a premise (e.g. urgent need of X) is not always explicitely available to the system and, hence, is designated as abducible. The rules with labels $pr1(X)$ and $pr2(X)$ define **priority** arguments. The claim of these rules is the *prefer/2* predicate and its arguments are rule labels, where the first has priority over the second. These rules can also have (non) defeasible premises.

The strength that arguments receive by priority arguments determines whether they can attack or defend against another argument. Informally, an argument is allowed to attack another argument only if it is at least as strong as the argument that it is attacking. The strength relation between two composite arguments is induced from the local strength relation that the priority rules, contained in the respective composite arguments, give to the other arguments contained in them. Informally, a composite argument, Δ_1, attacks another composite argument, Δ_2, whenever they are in conflict, and the arguments in Δ_1 are rendered by the priority arguments that it contains at least as strong as the arguments in Δ_2. In other words, if the priority arguments in Δ_2 render an argument in it stronger than an individual argument in Δ_1 then so do the priority arguments in Δ_1: a relative weak argument in Δ_1 is balanced by a weak argument in Δ_2.

In the example in Listing 1, let's consider that the user asks whether she should buy a bag when she knows that she needs it and that she is low on funds. Thus, *need(bag)* and *lowOnFunds* are non-defeasible facts. Argument rule *r1(bag)* (X instantiates to *bag*) supports the claim to *buy(bag)*. However, this is is potentially attacked by *r2(bag)* when $neg(urgentNeed(bag))$ holds. This is not a problem, as they attack each other, however, if *r2(bag)* is joined with *pr1(bag)*, which gives more relative strength to *r2(bag)*, then *r1(bag)* can only be admissible if it is joined with *pr2(bag)* to balance the relative strength. Thus, the argument $\Delta = [pr2(bag), r1(bag)]$ admissibly supports the claim to *buy(bag)*. Note that another way to defend against the potential attack by *r2(bag)* is to simply assume that $neg(urgentNeed(bag))$ does not hold i.e. there is another admissible set, namely $\Delta' = [ass(urgentNeed(bag)), r1(bag)]$, supporting *buy(bag)*.

2.2. Explainable AI

The central challenge facing **Explainable AI** is the generation of explanations that are both interpretable and complete (i.e. true to the computational model underlying the system). Depending on the complexity of the system, this trade-off between precision and interpretability of explanation becomes harder. Nevertheless, there are several properties of good quality explanations that we can follow, stemming mainly from the need for explanations to be cognitively compatible with the users and to be "socially useful" in clarifying the reasons underlying the results of the system that they are explaining [12]. Explanations need to be sensitive to the level of the user and to the purpose that the user needs the result of the system that is explained.

In general, there are three main cognitive and social requirements that form a good quality explanation. An explanation must be **attributive**, i.e. must give the basic and important reasons justifying why the result that it is explaining holds and could be chosen. Equally important is for the explanation to explain why the particular result is a good choice in relation to other possible results, i.e. the explanation must also be **contrastive**. Finally, a good explanation is also one that is **actionable**, i.e. where it is appropriate, it helps us understand what further actions we can take to confirm and to further build or utilize the result.

Systems built based on argumentation naturally lend themselves to being explainable due to the very close and direct link between argumentation and explainable justification (see e.g. [4, 5] for recent surveys on Argumentation and Explainable AI). Arguments supporting a claim, or conclusion, can provide the attributive part of an explanation, while the defending arguments, within an acceptable coalition of arguments resulting from the dialectic argumentation process to defend the attributive arguments against their counter-arguments, can provide the contrastive element of the explanation. These attributive and defending arguments also point towards taking (further) actions to confirm or question their premises, particularly when these relate to subjective beliefs or hypotheses.

For the particular case of the Gorgias argumentation framework, that we study here, this process of extracting natural explanations is facilitated by the way the admissible subsets of arguments that support a claim are constructed, and, eventually returned by the Gorgias system. As we have seen above, an admissible set of Gorgias contains, in general, object level arguments together with priority arguments at different levels. The object-level arguments would then give us the basic reasons supporting the claim whereas the priority arguments would give the reasons why to prefer this claim over the reasons that support alternative claims.

3. Explainable Gorgias Output

In this section we present in which way Gorgias Cloud aids the developer to validate or debug a policy using Application Level Explanations. This process will also help the reader understand how applications filter the Δ in order to show appropriate messages to the user, explaining the system's results.

For illustration purposes, let us consider a social media assistant agent and a particular user whose policy is the following:

Normally, give **default** priority to a post. If the topic of a post falls within the

> user's interests set the priority to **important**, unless the information of the post is negative. Posts that come from the user's manager are **important** regardless of whether they are positive or negative. **Hide**, posts that are on politics, unless the post is from the user's manager. **Hide** politics posts from the user's manager when negative, but when positive they are **important**.

In Listing 2 we can see the Gorgias code encoding the above personal policy by the social media assistant agent. Lines 1-6 define the three positions or options ($default(Post)$, $hide(Post)$ and $important(Post)$) as mutually exclusive using the *complement* predicate of the Gorgias framework.

The **object-level** rules in lines 7-9 state that all positions are possible for any new $Post$. Then, lines 10-25 define the **priority** rules for the user's policy. Specifically, lines 10-11 give default preference to rule with label $r1(Post)$, i.e. the default priority for any given $Post$. The $Post$ variable will be grounded to any new post. If the post is in the user's topics of interest then the priority is important (this is achieved by priority rules 12-14), unless the information of the post is negative (this is achieved in lines 15-16). If the post comes from the user's manager the priority is set to important, regardless if it is positive or negative (lines 17-19). Posts that are on politics are hidden (lines 20-23), unless the post is from the user's manager. In that case the post must be hidden, if it is negative, or marked as important, if it is positive (lines 24-25). Finally, each post can be assumed to be positive or negative if its status cannot be determined (note the **abducible** Gorgias predicate in lines 26-27).

For our running example, we can have a number of scenarios to test. One of them, "Test1.pl", is presented in Listing 3. In line 4 the reader can see a defeasible knowledge item, i.e. that the post with id *p1* is negative. The method to characterize a post as positive or negative, or assign a topic to it is out of the scope of this paper.

For the argumentation framework of Gorgias the correspondence between admissible coalitions of arguments and associated **explanations at the cognitive level of the application** can be constructed naturally from the internal Explanation, E, returned by the Gorgias system. We can automatically use the information (i.e. argument rule names) in E to construct an explanation, at the application level, that exhibits the desired characteristics of being *attributive, contrastive and actionable* as follows:

- **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.

These characteristics in the human-readable "Application Level Explanation" part of the output of the Gorgias Cloud IDE can be seen, for our example, in Listing 4. The attributive part of the explanation is in lines 8-9, while the contrastive part is in lines 10-12. In this example we do not have an actionable part.

If we consider the test file shown in Listing 5, the results are the ones shown in Listing 6. Here, besides the attributive (lines 8-9) and the contrastive (lines 10-13) parts we also find the actionable part in line 14. The supporting condition *negative(p2)* is provided by the abducible in line 28 of the theory (Listing 2). As the abducible represents an assumption, the action to verify it is suggested by the "Application Level Explanation" result.

Listing 2: The "socialMediaAssistant.pl" code.

```
complement(hide(Post),default(Post)).
complement(default(Post),hide(Post)).
complement(hide(Post),important(Post)).
complement(important(Post),hide(Post)).
complement(default(Post),important(Post)).
complement(important(Post),default(Post)).
rule(r1(Post),default(Post) ,[]):-newPost(Post).
rule(r2(Post),important(Post) ,[]):-newPost(Post).
rule(r3(Post),hide(Post) ,[]):-newPost(Post).
rule(pr1(Post),prefer(r1(Post),r2(Post)) ,[]).
rule(pr2(Post),prefer(r1(Post),r3(Post)) ,[]).
rule(pr3(Post),prefer(r2(Post),r1(Post)) ,[]):-my_topics(Post).
rule(c1(Post),prefer(pr3(Post),pr1(Post)) ,[]).
rule(pr4(Post),prefer(r2(Post),r3(Post)) ,[]):-my_topics(Post).
rule(c2(Post),prefer(pr1(Post),pr3(Post)),[negative(Post)]).
rule(d1(Post),prefer(c2(Post),c1(Post)) ,[]).
rule(pr5(Post),prefer(r2(Post),r1(Post)) ,[]):-manager(Post).
rule(c3(Post),prefer(pr5(Post),pr1(Post)) ,[]).
rule(pr6(Post),prefer(r2(Post),r3(Post)) ,[]):-manager(Post).
rule(pr7(Post),prefer(r3(Post),r1(Post)) ,[]):-politics(Post).
rule(c4(Post),prefer(pr7(Post),pr2(Post)) ,[]).
rule(pr8(Post),prefer(r3(Post),r2(Post)) ,[]):-politics(Post).
rule(c5(Post),prefer(pr8(Post),pr4(Post)) ,[]).
rule(c6(Post),prefer(pr8(Post),pr6(Post)),[negative(Post)]).
rule(c7(Post),prefer(pr6(Post),pr8(Post)),[positive(Post)]).
abducible(positive(_), []).
abducible(negative(_), []).
```

Listing 3: The "Test1.pl" testing file code.

```
newPost(p1).
manager(p1).
politics(p1).
rule(f1(p1),negative(p1) ,[]).
```

Let us summarize informally the automatic process of extracting an Application Level Explanation from the internal code Explanation that the Gorgias system returns. Such internal explanations contain the rule *labels* that were used to admissibly support the conclusion. For example, in Listing 4 the "Explanation" result is the $Explanation = [c4(p1), c6(p1), f1(p1), pr7(p1), pr8(p1), r3(p1)]$, representing the (composite) admissible argument for the given Position.

Initially, the priority rule labels, if any, are isolated in the $Explanation$ list. Moreover, for each priority rule of the list, the priority level is calculated. Next, the list is iterated and overlapping rules are deleted. That is, the lower priority rules in the rule priority hierarchy are deleted. Only high priority rules, from which object level arguments are derived, are kept in the list.

Listing 4: The hide(Post) query execution results for "Test1.pl".

```
prove([hide(Post)], Explanation).

Solution 1

Explanation=[c4(p1),c6(p1),f1(p1),pr7(p1),pr8(p1),r3(p1)],Post = p1

Application Level Explanation
The statement "hide(p1)" is supported by:
 - "newPost(p1)" and "politics(p1)" and "negative(p1)"
This reason is :
 - Stronger than the reason of "newPost(p1)" supporting "default(p1)"
 - Stronger than the reason of "newPost(p1)" and "manager(p1)" supporting "
     important(p1)" when "negative(p1)"
```

Listing 5: The Test2.pl testing file code.

```
newPost(p2).
my_topics(p2).
manager(p2).
politics(p2).
```

Listing 6: The hide(Post) query execution results for "Test2.pl".

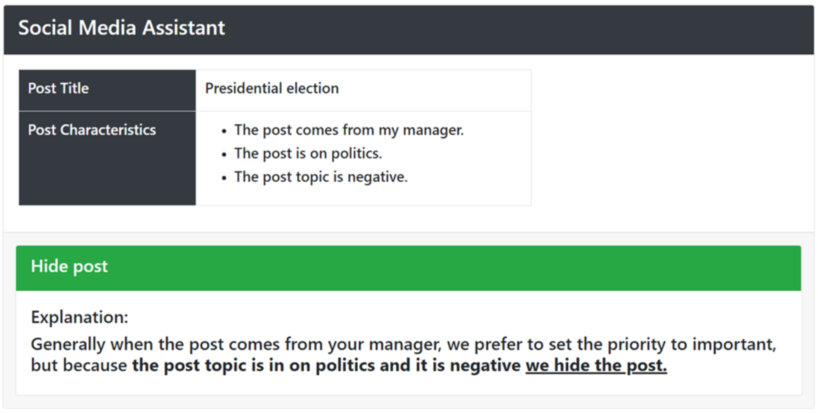
```
prove([hide(Post)], Explanation).

Solution 1

Explanation=[ass(negative(p2)),c4(p2),c5(p2),c6(p2),pr7(p2),pr8(p2),r3(p2)
    ], Post = p2

Application Level Explanation
The statement "hide(p2)" is supported by:
 - "newPost(p2)" and "politics(p2)" and "negative(p2)"
This reason is :
 - Stronger than the reason of "newPost(p2)" supporting "default(p2)"
 - Stronger than the reason of "newPost(p2)" and "my_topics(p2)" supporting
      "important(p2)"
 - Stronger than the reason of "newPost(p2)" and "manager(p2)" supporting "
     important(p2)" when "negative(p2)"
The supporting condition: "negative(p2)" is an assumption and needs to be
    confirmed.
```

Then the list is sorted in an ascending order. For each high level priority rule that is in the list: We recursively iterate the priority rules that lead to the weaker option, of the specific high level priority rule. The recursive iteration ends when the weaker object level rule is reached. During the recursive iteration of the high level priority list, a new list is created which contains:


Social Media Assistant

Post Title	Presidential election
Post Characteristics	• The post comes from my manager. • The post is on politics. • The post topic is negative.

Hide post

Explanation:
Generally when the post comes from your manager, we prefer to set the priority to important, but because **the post topic is in on politics and it is negative** **we hide the post.**

Figure 1: An example of explanation of a social media assistant.

- The weaker option for each high level priority rule,
- The weaker claim supporting facts,
- The stronger conclusion/position supporting facts,
- The high level priority rule, supporting arguments, that hold against the weaker option,

Finally, the supported Position is presented, along with its supporting arguments. Furthermore, for each overridden Position, its supporting information is presented, along with the information that caused the supported Position to be preferred over the overridden one.

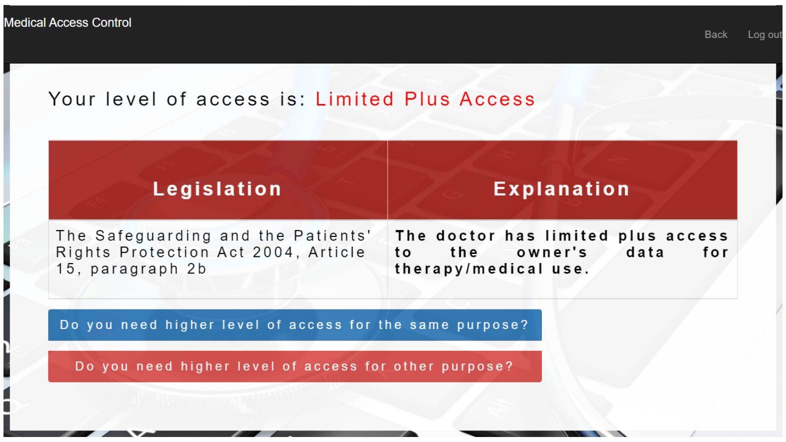
4. Explanations customized to the Application

In this section we showcase how the application level explanations that we can generate from the Gorgias internal explanation can be customized to the particular application at hand and thus provide natural and informative explanations to the user. We first show in Figure 1, an example of an explanation that a social media assistant, like the one we described in the previous section, would give to justify it's decision to hide a post. We then show examples from two other domain applications, namely that of (a) compliant Data Access to patient records and (b) Medical Decision support.

4.1. Explanations in compliant Data Access

An important recent field of application is that of data sharing. When different stakeholders want to exchange and share data, they need to agree on a common policy, usually referred to as a data sharing agreement (DSA). The Medica [7] system was developed for the purpose of determining the level of access to patients' data records, according to an EU country's law on medical data access.

There are six different access levels, ranging from *full access* to various types of *limited access* and to *restricted* or *no access*. The legislation recognized eight different persons' roles that are likely to request access ranging from the owner to his/her family members, to the family doctor,


Medical Access Control

Back Log out

Your level of access is: Limited Plus Access

Legislation	Explanation
The Safeguarding and the Patients' Rights Protection Act 2004, Article 15, paragraph 2b	The doctor has limited plus access to the owner's data for therapy/medical use.

Do you need higher level of access for the same purpose?
Do you need higher level of access for other purpose?

Figure 2: Attributive explanation of a data access level request in Medica.

to medical personnel, to third parties. Then, the purpose of the use of the data played a role on the access type and this, again, ranged from treatment to research, to processing, etc. Finally, specific contexts such as the possible consent of the owner or the fact that he/she may be dead decreed the access type.

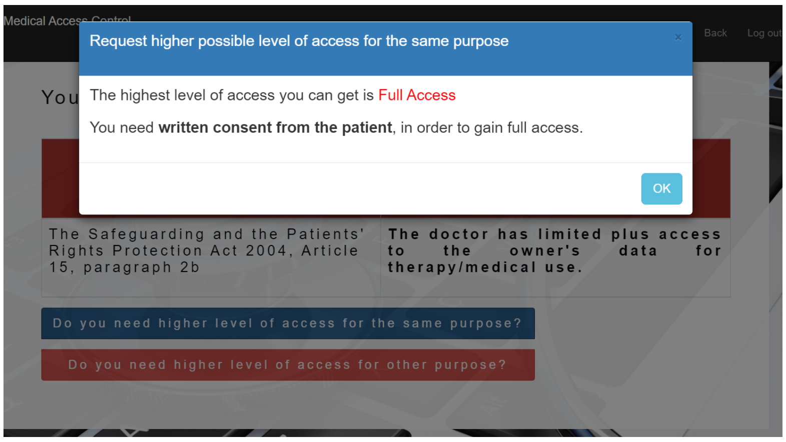
All decisions on the level of access reached by the $MEDICA$ system are explained to the user by reference to the relevant articles of legislation, which are, in fact, the basis for the object and priority level arguments that support the reached decision. Thus, the user sees attributive information for the proposed level of access. The system also supports requests from users wanting to gain higher level of access by indicating the extra information (the actionable part) that needs to hold, when indeed such higher level access is possible.

To illustrate the explanations in Medica let's consider the example in Figure 2. This screen comes after the request of a medical doctor for a patient's blood test for the purpose of treating the patient's situation. The hospital clerk can see two explanations based on the signatures that appeared in the Δ returned by the system. There are two explanations per signature, one that is free text so that the clerk understands clearly what is the suggested course of action. The other is a reference to the competent law article and paragraph so that a requester who wants to argue can have a basis for a rebuttal.

Medica allows the requester to ask for information, either how to get a better level of access for the same reason, or for another reason. At this time the system works with another version of the decision theory that has abducibles either the purposes of access or the specific contexts that can arise. Then it provides the actionable information to validate abducibles that lead to the desired outcome (see Figure 3).

4.2. Explanations in Medical Diagnostic Support

In this section we consider a medical decision support system and show how the same Gorgias Internal Explanations are unravelled to provide appropriate explanations for the application at hand. This is a medical diagnostic system for real-life clinical support in the field of Gynecology.


Medical Access Control

Request higher possible level of access for the same purpose

The highest level of access you can get is Full Access

You need **written consent from the patient**, in order to gain full access.

OK

The Safeguarding and the Patients' Rights Protection Act 2004, Article 15, paragraph 2b

The doctor has limited plus access to the owner's data for therapy/medical use.

Do you need higher level of access for the same purpose?

Do you need higher level of access for other purpose?

Figure 3: The actionable part in Medica comes after a user has expressed the desire to get another result.

The overall requirement of this system, called *GAID: Gynecological AI Diagnostic Assistant* [8], is to:

"Support clinicians to feel more confident in decision, helping to avoid over-diagnosis of common diseases and to ensure that emergency cases are not missed out".

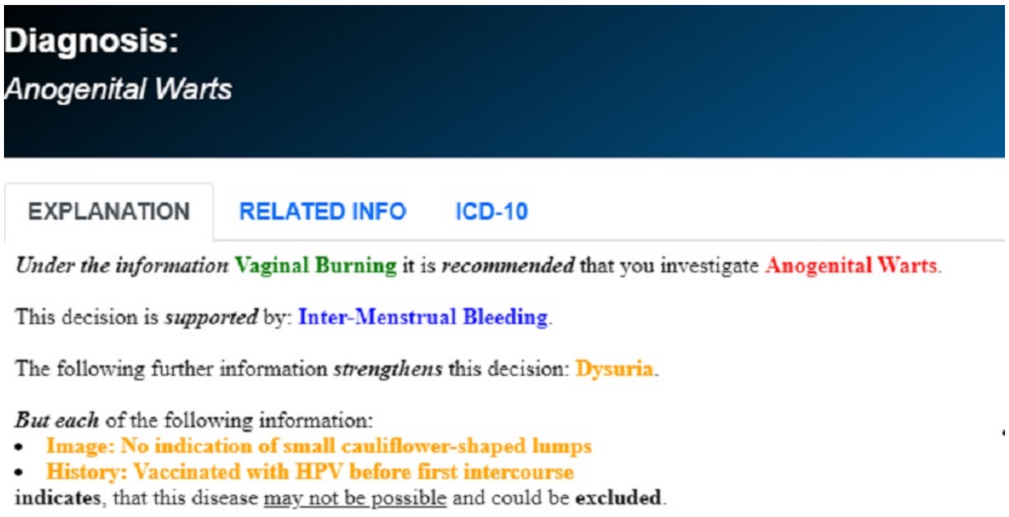
It is evident from this, that explanations would need to have a central role in building such a system. In effect, the approach taken is not to build a perfect or optimal diagnostic system (that could replace the doctor at some point) but rather to build a "peer" system that can "expertly" assist the doctors in their decisions. Hence, explanations are of paramount importance.

The system covers fully the area of Gynecology with 137 diseases (i.e. diagnostic decision options) and over a thousand different parameters (current symptoms, patient record, clinical examination findings and laboratory tests) that can affect the diagnosis. The knowledge on which the system builds its diagnosis was gathered using the SoDA methodology [13] and then captured within the Gorgias argumentation framework[2].

When using the system, the clinician enters information and the system offers a set of suspicious diseases as its diagnosis. When the clinician clicks on a suspicious disease, an explanation appears of why the disease is suspected. A typical explanation as presented to the user has the form shown in the example figure 4.

Hence the argumentative dialectic reasoning process is mapped onto "peer explanations" for the medical practitioners to understand the justification for a diseases to be suspicious or not. These explanations contain *attributive* reasons for a disease being suspicious, stemming from the basic arguments supporting the disease, as well as *contrastive* reasons of why this suspected disease may or may not be preferable over other diseases, stemming from the contextual strength arguments in the acceptable case for the disease. Explanations can also contain an *actionable* element, e.g. in the example above, it is suggested to investigate the information "Vaccinated with HPV" in order to accept the suggested disease as suspicious or not.

[2]The details of this process is beyond the scope of this paper.



Diagnosis:
Anogenital Warts

EXPLANATION RELATED INFO ICD-10

Under the information **Vaginal Burning** it is *recommended* that you investigate **Anogenital Warts**.

This decision is *supported* by: **Inter-Menstrual Bleeding**.

The following further information *strengthens* this decision: **Dysuria**.

But each of the following information:

- **Image: No indication of small cauliflower-shaped lumps**
- **History: Vaccinated with HPV before first intercourse**

indicates, that this disease may not be possible and could be **excluded**.

Figure 4: Example of Explanation of the $GAID$ Gynecology Assistant.

5. Conclusions

We have presented argumentation-based application level explanations for real-life domains and how the Gorgias framework supports explanations that are: a) **attributive** (explain why the Position is supported), b) **contrastive** (explain why the supported Position is preferred over other positions possible in the same given situation), c) **actionable** (propose that some information be further explored for its validity).

The emphasis is placed on the fact that these explanations are aimed to application developers or end-users, not computational argumentation scientists. Recently, in the work of [14], further tools have been developed that allow us to see in detail the dialectic argumentation process of Gorgias. These operate at the level of the internal explanation of the Gorgias system and could be used to further inform the application level explanations. In future we also plan to work on generating automated explanations based on the structured natural language text provided by the developers and/or the domain experts as requirements.

References

[1] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Computing Surveys 51 (2018).

[2] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information Fusion 58 (2020) 82–115. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103. doi:`https://doi.org/10.1016/j.inffus.2019.12.012`.

[3] European Commission, Regulation (EU) 2016/679 of the European Parliament and of the

Council of 27 April 2016, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[4] K. Čyras, A. Rago, E. Albini, P. Baroni, F. Toni, Argumentative XAI: A survey, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4392–4399. doi:10.24963/ijcai.2021/600.

[5] A. Vassiliades, N. Bassiliades, T. Patkos, Argumentation and explainable artificial intelligence: a survey, The Knowledge Engineering Review 36 (2021).

[6] A. C. Kakas, P. Moraitis, Argumentation based decision making for autonomous agents, in: The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS Proceedings, ACM, 2003, pp. 883–890. doi:10.1145/860575.860717.

[7] N. I. Spanoudakis, E. Constantinou, A. Koumi, A. C. Kakas, Modeling data access legislation with gorgias, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2017, pp. 317–327.

[8] P. Tanos, I. Yiangou, G. Prokopiou, A. Kakas, V. Tanos, Gynaecological artificial intelligence diagnostics (gaid). a pilot study of gaid and its performance., in: 31st Annual Congress of the European Society for Gynaecological Endoscopy, Lisbon, 2-5,October, 2022.

[9] A. C. Kakas, P. Mancarella, P. M. Dung, The acceptability semantics for logic programs, in: Proc. of 11th Int. Conf. on Logic Programming, 1994, pp. 504–519.

[10] A. C. Kakas, P. Moraitis, N. I. Spanoudakis, *GORGIAS*: Applying argumentation, Argument & Computation 10 (2019) 55–81. doi:10.3233/AAC-181006.

[11] N. I. Spanoudakis, G. Gligoris, A. C. Kakas, A. Koumi, Gorgias Cloud: On-line Explainable Argumentation, in: System demonstration at the 9th International Conference on Computational Models of Argument (COMMA 2022), 2022.

[12] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1–38. doi:10.1016/j.artint.2018.07.007.

[13] N. I. Spanoudakis, A. C. Kakas, P. Moraitis, Applications of Argumentation: The SoDA Methodology, in: ECAI, 2016, pp. 1722–1723. doi:10.3233/978-1-61499-672-9-1722.

[14] A. Vassiliades, I. Papadimitriou, N. Bassiliades, T. Patkos, Visual Gorgias: A Mechanism for the Visualization of an Argumentation Dialogue, in: 25th Pan-Hellenic Conference on Informatics, PCI 2021, ACM, 2021, p. 149–154. doi:10.1145/3503823.3503852.