Argument & Computation 0 (2018) 1–0
IOS Press

GORGIAS: Applying Argumentation

Antonis C. Kakas [a,*], Pavlos Moraitis [b], Nikolaos I. Spanoudakis [c]
[a] *Department of Computer Science, University of Cyprus, Cyprus*
E-mail: antonis@ucy.ac.cy
[b] *LIPADE, Paris Descartes University, France*
E-mail: pavlos@mi.parisdescartes.fr
[c] *School of Production Engineering and Management, Technical University of Crete, Greece*
E-mail: nikos@amcl.tuc.gr

Abstract. This paper presents the past and present efforts of developing real-life applications of argumentation within the preference-based structured argumentation framework of *Logic Programming with Priorities* and its *Gorgias* system. Since its free availability on the web in 2003, the *Gorgias* system has been used by different groups in a variety of real-life applications in areas such as medical support, network security, business computing, ambient intelligence and recently in the area of cognitive personal assistants. We will briefly review the *Gorgias* framework and its past applications and present an emerging general methodological approach for developing "decision making" applications of argumentation. This approach allows the development of real world applications directly from their high-level requirements expressed in the language of the application without the need for familiarity with the underlying technicalities of argumentation. A new tool, called *Gorgias-B*, supports this high-level development of applications and automatically generates the underlying argumentation theory *Gorgias* code. The paper will also report on ongoing real-life applications in two domains: an eye-clinic assistant for first level support, and patient data access and sharing agreements according to the relevant legislation and contracts or policies of several stakeholders involved. The proposed approach is quite general for this type of applications of argumentation and hence can be used more widely with any preference-based argumentation framework.

Keywords: Artificial Intelligence, Argumentation, Applications

1. Introduction

The natural link of Argumentation to Human Reasoning (see e.g. [1]) makes it an appropriate framework for applications in Artificial Intelligence (AI). This suitability of argumentation applies both to the realm of problems with strict specifications requiring expert (e.g. scientific) reasoning as well as to the more recent area of Cognitive Computing where the reasoning required is closer to that of common sense used by people at large in their everyday tasks.

In today's era of human-like AI, there are three main challenges to applications, which are particularly well served by an approach based on argumentation. Firstly, applications need to be built at a high-level elicitation or acquisition of their requirements akin to the natural cognitive level of the domain experts or personal users for whom the application software is built. Secondly, these systems need to be incrementally developed and improved or adapted to new and changing requirements in a highly modular way, possibly through a continuous learning process that accompanies development. Finally, systems need to be able to explain their decisions to human users (a requirement that soon will be required by law

*Corresponding author. E-mail: antonis@ucy.ac.cy.

in the European Union) and argue about their position and the suitability of their operation with respect to the given domain knowledge and the high-level operation guidelines provided by their owners/users.

Argumentation has experienced an ever increasing interest in its application to practical problems of AI. These problems range over a wide spectrum with applications focusing on the analysis of debates (e.g. debates in on-line social interaction [2–4]) to problems where a formal structure of argumentation needs to be computationally captured (e.g. in automating legislation [5, 6]). There are also several systems of argumentation CaSAPI [7], DeLP [8], TOAST [9] and Gorgias [10] that provide tool support for studying various aspects of computational argumentation.

A wide class of applications of argumentation concerns decision problems in dynamic and incomplete or uncertain environments. These include diagnosis problems, e.g. in the medical domain [11, 12], or in other domains such as home networks [13]), legal problems (e.g. [14]), or recommendation problems (e.g. medical treatment recommendation as in [15]). More generally, some practical works in the category of "decision making" applications include finding interesting products in e-commerce [16], negotiating supply strategies [17], making credit assignments [18], managing waste-water discharges [19], deciding about an automatic freight process [20], improving the performance of transport systems in rural areas [21], emergency rescue [22], aggregating clinical evidence [23], smarter electricity [24] and delivering clinical decision support services[1] [25]. Nevertheless, there is lack of work that would help the systematic development of argumentation software for real world applications.

This paper aims to address this gap by proposing a simple yet powerful methodological approach that can facilitate the systematic development of (a certain class of) applications of argumentation. The aim is to allow experts in their application domains to use argumentation based methods and tools for modeling their decision making problems with no or little need for technical knowledge on formal argumentation. This approach has emerged out of the experience from several real-life applications of argumentation based on the preference-based structured argumentation framework of *Logic Programming with Priorities* [10, 26, 27] and its *Gorgias*[2] system of implementation. These include applications in medical support systems, network security, business computing, ambient intelligence, pervasive services and, recently, cognitive personal assistants.

The approach is based on the consideration of application scenarios and the expression, by the domain expert, of statements of preference on the possible solutions within these scenarios. The expert is guided to analyze the conflicts in the application scenarios or combinations of these and to consider possible refinements of the scenarios that could possibly resolve or mitigate conflict. The important characteristic of the approach is that it can be carried out at a high level familiar to the domain expert. The approach abstracts away from the argumentation technicalities through the possibility of automatically transforming the high-level specification into an argumentation theory with an executable *Gorgias* code. As such the approach aims to address, at least partly, the three challenges for today's AI systems mentioned above.

The paper will briefly review the preference-based argumentation framework of *Gorgias* together with various real-life applications of argumentation based on it. The methodological approach for developing applications of argumentation, that has emerged through this variety of applications, is presented and illustrated through some of these applications and other examples. We show how a new human-machine interface tool, called *Gorgias-B*, supports the proposed approach and provides an automatic translation of the high-level scenario-based preference specification into a corresponding argumentation theory and

[1] http://www.openclinical.org/argumentation.html

[2] http://www.cs.ucy.ac.cy/ñkd/gorgias/

Gorgias software code. The paper also reports on an ongoing real-life application of an eye-clinic assistant for providing first level support at the clinic by ascertaining the severity of the case of a new patient and thus helping in prioritizing patient appointments. Similarly, we report on the current development of data access and sharing applications for patient records where multiple policy requirements from legislation and other stakeholders need to be integrated together to regulate information access.

In section two we present the general class of application problems for argumentation that we will consider in this paper. We describe the overall structure of these problems and illustrate this with an example. Then, in section three, we show how these problems can be captured within the argumentation framework of *Gorgias*. In section four we discuss the various real-life applications developed with *Gorgias*. Subsequently, in section five, we present the general knowledge engineering approach for applications of argumentation, along with the *Gorgias-B* tool developed for supporting it. Section six highlights the current challenges for *Gorgias* by providing an insight of the real-life applications under development and section seven concludes.

2. Application problems for Argumentation

A wide class of application problems which can be captured within argumentation are in essence *decision problems*. Argumentation is particularly suitable when the application environment is incomplete and dynamic. The incompleteness and changing information typically means that there is insufficient information to have a strict decision for a solution and hence the flexibility of argumentation provides a way to account for and manage the possible alternative solutions.

In this paper we will consider applications which can be formulated in the form of a decision problem whose language of description is composed of the following parts:

OPTIONS: comprising a set of the various results of the decision making problem. Options characterize the solutions to the application problem. In practice options are typically actions and hence the problem is to decide what course of action to take, e.g. to decide on a user's level of access to a sensitive data item. In some cases, options are explanations that help to classify a situation, e.g. to match the symptoms of a patient with a set of possible diseases. In many of these cases the explanation is just an intermediate step towards achieving the overall objective to decide on a course of action, such as what therapy to administer.

SCENARIO INFORMATION: comprising a set of relations that are used to (partially) describe the state of the application environment. These capture the various types of information that we expect to be available from the application environment when we are asked to apply the decision making to solve particular instances of the problem. This information can be sensory information readily available every time that we apply the decision making to solve a problem or it may be information that can be actively sought from the environment. In this case the application system may need to prompt the environment for this information. This practical consideration of the sensory information separates this into two types: *directly observable* and *hypothetical* with the latter requiring a decision to actively seek it.

SCENARIO-Based PREFERENCES: comprising a set (table) of tuples $\langle S\,;O\rangle$ of (partial) scenarios, S, together with a corresponding preferred subset, O, of options. This part of the description of an application captures the criteria or guidelines under which the solutions of the problem should be sought. They can be seen as high-level requirements to help find a satisfactory solution in

any particular circumstance of the problem given by some partial description of the application environment. Note that the scenarios S that appear in these statements refer to a minimal set of information about the application environment that is sufficient for the domain expert to be able to express some preference amongst the possible options. As we will see below it is very useful to group these scenarios in hierarchies of increasing specificity.

It is important to note that this "language scheme" for the decision problems allows the application domain experts to formulate them with their familiar language and to describe their application at a high and non-technical level. The domain expert does not need to be familiar with the technicalities of the underlying argumentation framework in which their problem will eventually be expressed.

As mentioned above, the third component, that of Scenario-based Preferences (SP), captures at a high level the required behaviour of any system for solving the problem. Depending on the application these are expressions of expert knowledge, e.g. in a medical support application the expert (doctor or other medical personnel) defines the set of preferred possible causes, or actions to be taken, for various patient scenarios. On the other hand, for human-like AI applications, such as cognitive assistants, the Scenario-based Preferences come from personal preferences of the human user. These can be in the form of high-level guidelines for the assistant. Consider, for example, the preference for good quality food or the avoidance of red meat, in the context of building on-line shopping personal assistants. As we will see below, this "language scheme" is indeed suitable for the formulation of a wide and varied class of application problems.

In order to illustrate this high-level description of an application problem let us consider a simple example where we want to capture the guidelines of a human user for an on-line food shopping personal assistant. The set of options in this problem is to buy or not to buy various products in a supermarket. For simplicity, let us assume that this set contains the following options (and their negations) :
$OPTIONS = \{buy(lamb), buy(pork), buy(chicken), buy(fish)\}$

The decision problem is to decide which *buy* action to select. For ease of presentation, we assume that we only buy one type of these foods when shopping, i.e. that these options are mutually exclusive, although as we will see the framework will allow this to be a "soft" requirement. The user has informed us that all these options are enabled under the minimal scenario information of the "main shopping" for the week (denoted here by *main_shopping*). This can be captured by a **basic or initial** scenario-based preference statement such as:
$SP_0 = \langle S_0 = \{main\_shopping\} \; ; \; O_0 = \{buy(lamb), buy(pork), buy(chicken), buy(fish)\}\rangle$

We will say that these options are **enabled** or are **available** by this basic underlying scenario S_0. The user can then express preferences on these enabled options depending on different scenario information that is indeed sufficient for a meaningful preference to be expressed. For example, we can have:
$SP_1 = \langle S_1 = \{main\_shopping\} \cup Cheaper(Foods) \; ; \; O_1 = Buy(Foods)\rangle$
expressing the preference for cheaper food. Note that the set $O_1 = Buy(Foods)$ may contain several options as several different foods can be cheap at the same time. Let us assume that typically pork and chicken are both cheaper forms of food (amongst the ones we are considering here), i.e. $Cheaper(Foods) = \{cheap(pork), cheap(chicken)\}$ and hence $O_1 = \{buy(pork), buy(chicken)\}$.

The user may have a preference amongst the cheaper options, e.g. a preference for pork in the winter and for chicken in the summer. This additional preference may be captured in further scenario-based preferences whose scenarios are **refinements** of the scenario S_1, in the same way that S_1 is a refinement of the initial scenario S_0:
$SP_1^1 = \langle S_1^1 = S_1 \cup \{winter\} \; ; \; O_1^1 = \{buy(pork)\}\rangle$

$SP_1^2 = \langle S_1^2 = S_1 \cup \{summer\} \; ; O_2^2 = \{buy(chicken)\}\rangle$

In refinements of scenarios the user is able to **focus** further her/his preference amongst the preferred options of the parent scenario. Note that given a scenario the user may have different independent ways to refine the scenario. In our running example, the user may have another preference amongst the cheaper options, e.g. for the foods that are locally produced. Hence we have a new refined scenario-based preference of:

$SP_1^3 = \langle S_1^3 = S_1 \cup Local(Foods) \; ; O_1^3 = Buy(Foods)\rangle$

where in O_1^3 we would have locally produced the cheap foods, e.g. $O_1^3 = \{buy(chicken)\}$.

When we have different scenarios whose conditions can hold together in the application then we are led to consider **combinations** of scenarios. We consider a new scenario made up of the union of the two scenarios. The user considers what are the preferred options amongst the union of the preferred options in the two scenarios. This occurs for example when we have different refinements of a scenario that can occur together, as is the case here with scenario S_1 where both the refinements of *winter* (or *summer*) and $local(chichen)$ can hold together :

$SP_1^{13} = \langle S_1^{13} = S_1^1 \cup S_1^3 = S_1 \cup \{winter\} \cup \{local(chichen)\} \; ; O_1^{13} = \{buy(chicken)\}\rangle$,

which expresses the preference for the locally produced chicken even in the winter time, where pork is generally preferred amongst the cheap options.

In our example, we have considered so far refinements and combinations of scenarios based on the scenario of "cheapness of price". We could have other scenario conditions on the application environment that are independently sufficient to express a preference. For example, the preference for locally produced foods of the user may be a general preference that applies irrespective of the price. If this is the case this general preference is not captured by the scenario-based preference, SP_1^3 above, as this applies as a preference only amongst the cheap options and not as an overall preference for locally produced food. If we wanted the later we would have a new scenario-based preference based on a new independent refinement of the initial scenario, S_0,:

$SP_2 = \langle S_2 = S_0 \cup Local(Foods) \; ; O_2 = Buy(Foods)\rangle$,

where the preferred options O_2 will contain the options of buying the different locally produced foods, e.g. $O_2 = \{buy(chicken), buy(lamb)\}$ when only chicken and lamb are locally produced.

Similarly, we would consider new independent scenarios if the user's "seasonal preference" for pork in the winter and fish in the summer was general irrespective of price. This then would also give us new scenario-based preferences such as:

$SP_3 = \langle S_3 = S_0 \cup \{winter\} \; ; O_3 = \{buy(pork), buy(lamb)\}\rangle$

$SP_4 = \langle S_4 = S_0 \cup \{summer\} \; ; O_4 = \{buy(fish), buy(chicken)\}\rangle$

Given such independent scenario based preferences we are led to consider their possible **combinations** to see if indeed a combined scenario can occur in the application environment, i.e. that the independent scenario conditions are not also mutually exclusive, and when this is the case to find from the user if there is a preference amongst the union of the preferred options of the individual scenarios that we are combining.

For example, if we consider the scenario-based preferences, SP_1 and SP_2 above, we can see that these do not express a preference between cheap and local foods. In the combined scenario, S_{12}, where we have both conditions of $Cheaper(Foods)$ and $Local(Foods)$ we can now ask or learn from the user if s/he has a preference pertaining to such a combined situation, e.g. a preference between pork, chicken and lamb in our example. We would then have a new scenario-based preference:

$SP_{12} = \langle S_1 \cup S_2 \; ; O_{12}\rangle$,

where the subset O_{12} contains such a preference, e.g. $O_{12} = \{buy(chicken)\}$, for the locally produced chicken, in our example recovering the preference that we got earlier with the refinement scenario of SP_1^3. Note that the difference of having SP_{12} instead of SP_1^3 is that now when we do not have any information about cheaper foods we will still have some preference for the locally produced food.

Similarly, we would consider combinations of the scenario S_1 with S_3 and S_4 which will recover the preference statements in the refined scenarios of SP_1^1 and SP_1^2 respectively. We can also consider combinations of the scenario S_2 with S_3 and S_4 to give us new scenario-based preferences between local produce and the general seasonal preferences of our user. Of course the combination of scenarios S_3 and S_4 is not possible and hence it is not considered for further preference statements. Note that once we are given a new preference in a combined scenario we can continue to consider refinements or further combinations of this combined scenario. In certain applications, it is possible that in the combined scenario the list of preferred options also contains options that are outside the union of the individual scenarios that were combined. For example the user may prefer lamb for special occasions, chicken when s/he is not feeling well but when both these hold (i.e. a special occasion when s/he is not feeling well) s/he prefers fish.

We can also have "local" preference statements between an individual option and its negation. For example, the user may express the preference not to buy fish if it is farmed:
$SP_{ff} = \langle S_{ff} = \{farm(fish)\} ; O_{ff} = \{not(buy(fish))\}\rangle$

These are "vertical" preferences that apply locally only to the particular option involved.

Finally, we note that some of the scenario information may not be readily available to the shopping assistant when this is operating, e.g. in our example the user may have expressed a preference for buying foods (e.g. for lamb or fish) when the week contains a special occasion but this information (of *special_occassion*) is not always directly available to the assistant. When this is so we can designate such conditions as **hypothetical** so that when this information is indeed not readily available to the system this would be able to perform an action (e.g. prompt the user) to find out about this. For example, if in some week the lamb is on special offer and hence a cheap food for that week the system, by asking and finding if there is a special occasion in this week, could then have a preference to buy lamb.

A principled approach to capture the preference requirements or guidelines of an application problem would involve identifying possible combinations of those scenarios expressed directly by the user, that contain conflicting preferences and prompting or learning from the user further preferences under these combined scenarios and/or their refinements. This process would be applied iteratively to consider further combinations with any new scenarios introduced. In section 5 we will present a general methodological approach for eliciting from the application domain experts these preference requirements.

3. Applications in the Gorgias Argumentation Framework

In any application of argumentation solving a problem essentially amounts to finding solutions, e.g. options, that can be supported by a **good quality** or **acceptable** argument. This provides the important link between applications and computational argumentation: formulate the problem in an argumentation setting and then look for acceptable arguments. It is therefore important to be able to generate from an application problem specification, as presented above, a corresponding argumentation theory.

In this section, we will overview briefly the Gorgias argumentation framework and indicate how application problems, described as above in terms of scenario-based preferences, can be formulated as an argumentation theory in this framework.

The Gorgias framework is a structured argumentation framework where arguments are constructed using a basic argument scheme of Modus Ponens to link a set of premises with the claim or position of the argument. We will denote an **argument**, A, by $A = Premises \triangleright Claim$ representing a set of **argument rules** that together with the $Premises$, given as facts, the $Claim$ is derived via successive application of Modus Ponens. These argument rules have the syntax of Extended Logic Programming, i.e. rules whose conditions and conclusion are positive or explicit negative atomic statements without the need to use a second form of Negation as Failure in these conditions[3]. The claim of an argument can be a positive or negative atomic statement.

In the context of the type of applications, as described above in this paper, the premises are typically given by a set of conditions describing a scenario and the claim is an option. Furthermore, in applications where the scenario conditions are themselves defeasible, i.e. they also constitute **beliefs** which can be argued for or against, the claim of an argument can also be a literal on a belief scenario predicate. The distinction, then, between beliefs and options, is that options cannot be premises of arguments whose claim is a belief.

When the claim of an argument is a literal on an option (or belief) predicate, this argument is called an **object-level** argument. These object-level arguments in the framework are supplemented by **priority arguments** whose purpose is to give a relative strength between arguments and thus induce an **attack relation** between arguments and sets of arguments. The priority arguments express a local preference between two arguments. They have the same syntactic form of $P = Premises \triangleright Claim$, but now the $Claim$ is of a special form, $a_1 > a_2$, where a_1 and a_2 are any two other individual argument rules. Note that a_1 and a_2 can themselves be priority argument rules, in which case we say that the argument, P, is a **higher-order** priority argument expressing the fact that we prefer one priority over another, in the context of the premises of P. Hence, the framework of Gorgias is a preference-based argumentation framework where we can express **conditional** preferences and **higher-order** preferences over arguments.

The formulation of an (application) problem is then given by a **Gorgias argumentation theory** where the knowledge describing the application is represented in terms of object and priority arguments rules. The dialectic argumentation process to determine the acceptability of an argument supporting a desirable claim (e.g. an option) occurs between **composite arguments** comprised of a (minimal and closed) set of arguments, $\Delta = (A_O, A_P)$, containing a subset, A_O, of object level argument rules and a subset, A_P, of priority argument rules, chosen from the given argumentation theory. These composite arguments form the set, $Args$, of (formal) arguments in a corresponding abstract argumentation framework $\langle Args, ATT \rangle$.

The attack relation, ATT, between two composite arguments is induced from a notion of strength that the priority arguments give to the arguments that they accompany inside a composite argument. Informally, a composite argument, Δ_1, attacks another composite argument, Δ_2, whenever they are in conflict, i.e. they support incompatible claims or are based on incompatible premises, and the arguments in Δ_1 are rendered by the priority arguments that it contains at least as strong as the arguments contained in Δ_2. In other words, if the priority arguments in Δ_2 render an argument in it preferred over 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. The precise detail of the definition of this attack relation is not important in this paper and can be found in [10, 28].

Once we have such a corresponding abstract argumentation framework, $\langle Args, ATT \rangle$, we can then use one of the standard definitions of acceptability of arguments to select the acceptable arguments within

[3]There are historical reasons for this as the original motivation of presenting the Gorgias argumentation framework in [28] was to give an argumentation formulation for Logic Programming without Negation as Failure.

the argumentation theory that represents our application problem and thus get to "good" solutions to our problem, i.e. solutions that conform to the guidelines of the problem representation expressed in the argumentation theory. These solutions will be the options supported by the acceptable arguments. In the special case where there are acceptable arguments supporting only one of the possible options in the problem this option is regarded as a best or optimal solution. In the Gorgias framework the semantics for the acceptability of (composite) arguments used is that of **admissibility**, namely (composite) arguments that are not self-attacking and they attack back all other arguments that attack them.

Let us illustrate the Gorgias argumentation framework and how we can capture a problem specification in terms of an argumentation theory using the simple example of the on-line cognitive shopping assistant (partially) specified above.

The knowledge of the various minimal (possibly empty) set of scenario conditions, S_0, in which options are specified as possible ones is represented by object-level arguments, $a_i = S_0 \rhd O_i$ for each option O_i in the preferred subset of options, P_0, in the corresponding scenario-based statement, $SP_0 = \langle S_0; P_0 \rangle$. Hence, in our example above, with $S_0 = \{main\_shopping\}$, this would give[4] a set of object-level arguments represented by:
$a(Food) = \{main\_shopping\} \rhd buy(Food)$,

which, given the four values of the variable parameter, $Food$, includes the four arguments:
$a(pork) = \{main\_shopping\} \rhd buy(pork)$
$a(lamb) = \{main\_shopping\} \rhd buy(lamb)$
$a(chicken) = \{main\_shopping\} \rhd buy(chicken)$
$a(fish) = \{main\_shopping\} \rhd buy(fish)$

These arguments enable their corresponding options. Other such minimal scenarios can exist for enabling options, e.g. when some foods are on special offer. This would introduce additional object-level arguments, such as:
$a_{so}(Food) = \{special\_offer(Food)\} \rhd buy(Food)$,

giving one argument for every value of *Food* that is on special offer.

Then given any further scenario-based preference statements we can build priority arguments on top of these object-level arguments to capture these statements. For example, SP_1 in our example that expresses the general preference for the cheap foods would be represented by the priority argument rules:
$p(Food) = \{Food \in Cheaper(Foods), OFood \notin Cheaper(Foods)\} \rhd (a(Food) > a(OFood))$
capturing a set of specific priority arguments for the current cheap foods, such as for example the preference of pork and chicken over lamb and fish when indeed these are cheaper than these:
$p_{c1}(pork) = \{cheap(pork)\} \rhd (a(pork) > a(lamb))$
$p_{c2}(pork) = \{cheap(pork)\} \rhd (a(pork) > a(fish))$.
$p_{c1}(chicken) = \{cheap(chicken)\} \rhd (a(chicken) > a(lamb))$
$p_{c2}(chicken) = \{cheap(chicken)\} \rhd (a(chicken) > a(fish))$.
$p_{c3}(pork) = \{cheap(pork)\} \rhd (a(pork) > a(chicken))$
$p_{c3}(chicken) = \{cheap(chicken)\} \rhd (a(chicken) > a(pork))$.

Note that the last two priority rules capture the fact that the options of pork and chicken in SP_1 are no-comparable or equally preferred.

The further refined scenario-based preferences of SP_1^1 and SP_1^2 for *pork* in the winter and *chicken* in the summer will be represented by the additional *higher-level* priority argument rules:
$c_w(pork) = \{winter\} \rhd (p_{c3}(pork) > p_{c3}(chicken))$.

[4]Note that in the implementation language of Gorgias these are written as rules with $\rhd$ replaced by rule implication $\rightarrow$.

$c_s(chicken) = \{summer\} \rhd (p_{c3}(chicken) > p_{c3}(pork))$.

These higher level rules, by giving to the respective priority rules higher priority when the seasonal condition holds, they capture the implicit preference for more specific scenario-based preferences.

Similarly, the other refined scenario-based preference, SP_1^3, for a preference to local products amongst the cheaper foods will be represented by higher-level priority argument rules of the same form. For example, when chicken are produced locally we will have:
$c_l(chicken) = \{local(chicken)\} \rhd (p_{c3}(chicken) > p_{c3}(pork))$

The more refined scenario preference given by SP_1^{13}, which considers the combination of the "seasonal" and "local" refinements, will be captured at a yet higher-level priority argument rule:
$d(chicken) = \{\} \rhd (c_l(chicken) > c_w(pork))$,

expressing the fact that the priority given to local food is stronger than the (default) priority given to seasonally preferred food.

Finally, to capture the local preference of not buying fish when farmed (in SP_{ff}) we would have an object-level argument for the l=option not to buy fish:
$a(n\_fish) = \{farmed(fish)\} \rhd \neg buy(fish)$,

together with the priority argument rule:
$p_{nf}(n\_fish) = \{\} \rhd (a(n\_fish) > a(fish))$.

We therefore see that we can develop a systematic translation of scenario-based preferences where successive refinements of a scenario would give priority argument rules at a higher level every time we consider a further refinement in the scenario. These **levels of priority** in the argumentation theory that captures the problem specification, exploit the hierarchical argumentation framework encompassed by the framework of Gorgias. We can see this by examining how composite arguments are constructed from these individual arguments rules and how they attack each other.

Consider a current situation - a specific application scenario - where *main_shopping* holds, pork and chicken are cheaper than the other foods (lamb and fish) and it is winter. We can construct two composite arguments whose argument rules apply under the given application scenario:
$\Delta_1 = \{a(pork), p_{c1}(pork), p_{c2}(pork), p_{c3}(pork)\}$
$\Delta_2 = \{a(chicken), p_{c1}(chicken), p_{c2}(chicken), p_{c3}(chicken)\}$,

where the first one supports the option to buy pork and the second one the option to buy chicken. Each of these attacks arguments supporting the other two options of lamb and fish. For example, the argument $\Delta_3 = \{a(lamb)\}$ is attacked by Δ_1 as they are in conflict by supporting the incompatible options to buy pork and lamb. But Δ_3 does not attack Δ_1 because the priority rule $p_{c1}(pork)$ in Δ_1 makes its object-level argument $a(pork)$ strictly stronger than the conflicting object-level argument $a(lamb)$ in Δ_3. For Δ_3 to be able to attack Δ_1 it needs to be extended with a priority rule that would render its relative weak argument of $a(lamb)$ stronger than the conflicting argument of $a(pork)$. But in the current application scenario there is no such priority argument whose premise conditions hold and thus would be enabled to indeed give the higher relative strength to $a(lamb)$ (e.g. this would be the case if lamb was on special offer). Of course, Δ_3 attacks the simpler argument of $\{a(pork)\}$ but by including the priority arguments in Δ_1 this is no longer the case.

Between Δ_1 and Δ_2 the attack is symmetrical because they each include a priority rule that makes its arguments relatively stronger than the opposing one, i.e. Δ_1 contains $p_{c3}(pork)$ that makes its argument $a(pork)$ stronger than the conflicting argument $a(chicken)$ of Δ_2 and vice versa Δ_2 contains $p_{c3}(chicken)$ that makes its object-level argument relatively stronger. But Δ_1 can strengthen itself by also incorporating the priority argument rule $c_w(pork)$ which indeed applies in the specific application scenario since *winter* holds. To see how this strengthening of Δ_1 comes about we notice that Δ_1 and Δ_2 are

in conflict in a second way, namely that Δ_1 supports the preference of $a(pork)$ over $a(chicken)$ whereas Δ_2 supports the opposite of this. For this conflict, which comes from $p_{c3}(pork)$ and $p_{c3}(chicken)$, the extra priority rule $c_w(pork)$ gives relative priority to the first one and, hence, Δ_2 is attacked by the sub-argument $\{p_{c3}(pork), c_w(pork)\}$ but Δ_2 cannot attack back.

In fact, no composite argument supporting the option *chicken* can be constructed that attacks back all its attacking arguments and, hence, only the option of pork has *admissible* arguments and, hence, it is the only decision/solution that can be admitted in the specific application scenario. If, in addition, we are given or we find out that the chicken are locally produced then we can strengthen Δ_2 by adding $c_l(chicken)$ to enable this to attack back the attack by $\{p_{c3}(pork), c_w(pork)\}$. Then, by further adding the (higher-level) priority rule, $d(chicken)$, we make the new conflict between Δ_2 and Δ_1 on the priority amongst $p_{c3}(pork)$ and $p_{c3}(chicken)$ stronger for Δ_2, and, hence, Δ_1 is no longer admissible. Therefore, now the only admissibly supported option is to buy chicken.

If there are several options that are separately admissible in a partially given application scenario, then we either present the alternatives together with their supporting arguments or we can use a meta-policy that is also developed in the same way from scenario-based preferences given by the domain expert to further decide on what to choose. In our shopping example, we can buy both pork and chicken when the user specifies the meta-guideline to buy more than one meat/fish food item when it is affordable.

Summarizing, we see that once we have a description of an application problem in terms of scenario-based preferences, as described in the previous section, it is possible to generate a corresponding argumentation formulation with the Gorgias framework that faithfully captures these preferences. This is an important property of the argumentation-based formulation as it allows the development of applications at the level of discourse of the application domain experts without the need for them to be aware off any of the technical details of computational argumentation.

3.1. The Gorgias System

The Gorgias system was developed as a Prolog meta-interpreter to support the dialectical argumentation process of the framework described above. It was made publicly available on the web in 2003. The system supports queries to find which options are admissibly supported by an argumentation theory. Moreover, it provides the admissible composite arguments that support these options. It also supports hypothetical (abductive) reasoning, integrated with that of its dialectical argumentation, so that it can be used to generate further scenario information under which a desired option would be supported by an admissible argument and, hence, become a possible solution.

As we will see below this was used by several groups to develop applications in various domains. In addition, the Gorgias argumentation framework and its system formed the basis to study several general important problems in Artificial Intelligence such as non-monotonic learning [27], intra-agent control [29], multi-agent negotiation and dialogue [30, 31], distributed decision making [32], and, reasoning about actions and change [33]. In 2016 a new tool, called Gorgias-B, was released to help the development of applications of argumentation under Gorgias. This system will be described below in section 5.

4. Real-life Applications of Gorgias

In this section we present an overview of the various real-life applications problems that have been studied over the last decade or so within the Gorgias argumentation framework and where the Gorgias

system was used to implement and evaluate the argumentation-based approach to these problems. We will briefly catalog these past applications and then present in more detail one of these applications within the perspective of applications presented in section 2.

One of the first real-life applications of Gorgias [34] was in the area of Medical Informatics where the problem was to identify what medical actions, e.g. further medical tests or treatment, were needed to ascertain the seriousness of a patient with the possibility of Deep Venous Thrombosis (DVT). Medical expert knowledge was captured as a Gorgias argumentation theory in different agents for different expertise. The task was then to consider and weight up diagnoses for DVT or excluding DVT as well as the different further examinations on the patient that would help make a better diagnosis. Scenarios of the clinical history are considered and the argumentative agents deliberate between them. The Gorgias argumentation theories are role and context dependent and also contain priorities that are practically motivated. The application was build to provide support to medical practitioners and it was tested on a corpus of 600 patients at a collaborating hospital in Cluj-Napoca, Romania.

Gorgias has been applied in several problems within the general area of Ambient Intelligence. One such application in Ambient Assisted Living [35] concerned with providing home services for people suffering from cognitive problems and more particularly from Alzheimer disease. Agents reason with argumentation within the Gorgias system to arrive at decisions in situations where conflicting points of view exist, corresponding to doctors' opinions of different specialties, and where the patients may have multiple health problems. This work was part of a larger project, called HERA [36], which was successfully evaluated in real-life trials in two phases. The first phase took place at the Hygeia hospital[5] in Athens, Greece, and the second at the users' homes.

At the lower-level of challenges in ambient intelligence Gorgias was applied to address the problem of conflict resolution from networked sensors whose information is incompatible with each other [37]. This enables the development of context aware-pervasive services that can adapt suitably according to the application environment. This sensor's conflict resolution capability has been tested extensively using real-life Web services technology, capable of resolving conflicting data gathered from up to 10 sensors. The authors argue that this shows the potential of argumentation theory to solve real-world problems in services computing. Recently, we have also used Gorgias to address the problem of conflict resolution at the other end of the spectrum of resolving diplomatic disputes between countries [38]. Similarly, disputes can arise when we have a shared environment by many stakeholders where several legal and other contracts may apply. Again recently, in [39] Gorgias was used to address this in the context of Data Sharing of sensitive patient data.

Gorgias has also been use in applications of network security to provide support tools for authoring, analyzing and managing the firewalls for the security protection of a corporate network [40]. The management of firewalls becomes a challenging task as they grow through new requirements, imposed by the organization, and it is left to the experience of the administrators to address these new requirements by creating new firewall rules and inserting them at the best place inside the list of the existing rules. The overall aim of this application was to provide an automatic generation of firewall configurations from higher-level requirements and, thus, facilitate their authoring and maintenance. The argumentation-based formulation of firewalls allows a direct mapping from the high-level network security policy to the firewall policies. Under *Gorgias* we have an executable firewall configuration whose compliance to the policy is automatically ensured. This argumentation based approach was applied and evaluated with the firewall specification policy of a moderate-size enterprise, where it was used to examine the properties of

[5] http://www.hygeia.gr

the existing firewall and to test that it could automatically generate the relative rule orderings that would ensure the correctness of the firewall configuration, with respect to the given policy. It was shown that this was possible and that the performance of Gorgias was comparable, on specialized tasks, to state of the art approaches dedicated to network configuration management [41]. Recently, in [42], following a similar approach of formulating firewalls through argumentation, the authors provide a framework where explanations of the behaviour of the firewalls can be documented and exploited by the administrators and users of the network.

Another application was Market-Miner, an innovative agent for automated pricing mainly targeted for the retail sector [43]. Market-Miner captures the points of view of different departments of a firm (production, financial, marketing, etc) but also information coming from internal (results of data mining on the corporate database) or external sources (e.g. prices of the competition, weather forecast) and captures appropriate preferences in conflicting scenarios. Briefly, the options considered were to sell a product at a normal price, a high or a low price. Scenario-based preferences captured various company policies, e.g. a high pricing scenario when the object of the decision was a high technology product and an advertised invention, while a low pricing scenario when the product's type was within a category that the company wanted to hit the competition. The system was able to simulate the resulting prices under a specific strategy adopted by the firm and determine the profit margins that will come after such a move. This helped the organization plan its sales policy.

A recent application of *Gorgias* in a different area concerned a practical problem of image analysis, namely that of automated discrimination of handwritten/printed text [44]. This problem was modeled as a distributed decision making problem where the decision about the labeling of text (i.e. "handwritten" or "printed") is made through a bilateral dialogue between autonomous agents. *Gorgias* is used for implementing the decision making mechanisms of these agents. This provided a dynamic decision making process dealing with the possible dilemmas of the agents in conflicting situations, i.e. when both decisions "handwritten" or "printed" are admissible, using expert default (or generic) and contextual knowledge for solving these conflicts. Thus, each agent is able to propose in an dialogue with the other agent, a clear decision based on its own point of view, in order to look for a commonly accepted decision when they have an initial disagreement. The system has been evaluated on real-life data taken from the IAM handwriting database [6]. This system could be integrated in several real world applications such as printed text detection and extraction for Optical Character Recognition (OCR). It could also be used for improving the electronic document management systems (EDMS) allowing them to handle hybrid documents (e.g. printed documents with handwritten annotations).

4.1. Application in Investment Portfolio Construction

Typically, investors are concerned with constructing a portfolio of assets. One particular type of such assets is that of mutual funds. In these cases, a set of candidates is chosen, among which the investor constructs the portfolio according to personal preferences and forecasts. The different approaches applied in the literature, e.g. solving a linear programming problem, or apply multi-criteria decision aid methods (see for example [45, 46]) do not provide the opportunity to an individual investor to define different investment scenarios according to personal preferences, or take a decision in an environment with limited information, in order to select the best mutual funds which will compose the final portfolios. To take such a decision, the investor can take into account the figures and metrics extracted from

[6] http://www.iam.unibe.ch/fki/databases/iam-handwriting-database

previous year(s) performance of existing assets, the market condition, e.g. bull (rising) or bear (falling), that is always subject to forecast, and her/his personal attitude ranging from aggressive (preferring high risk assets that promise high returns) to defensive (preferring known successful assets with low risk).

In the Gorgias argumentation based approach for this problem [45] the central question was whether or not to add an asset to the investment portfolio. In an investment scenario it is possible to *select* or *not select* a *fund*. Thus, we consider the $OPTIONS = \{select(Fund), \neg select(Fund)\}$ and the development task is to capture a policy of requirements on the decision between these two options for any given $Fund$. The *scenario information* is information about the *Fund* typically extracted from the variables that measure the performance and risk of the MFs. This includes the following:

(1) return of the funds for the previous period
(2) standard deviation of the daily returns of a fund in the previous period
(3) the beta coefficient that computes a fund's risk in relation to the capital risk according to its sensitivity to fluctuations of the general financial market (its index)
(4) the Sharpe and Treynor indices for a fund's excess return based on a risk-free rate or investment.

Moreover, we get *scenario information* from the expected market condition, which can be unknown, or forecasted as *market(bear)* or *market(bull)*. Finally, the investor profile can be unknown or be identified as *investor(risk_averse)* or *investor(risk_tolerant)*. Thus, we can identify several contexts, e.g. two types of investors (risk averse and risk tolerant), a performance option (selection of funds with high Sharpe and Treynor indices), and two market contexts (bull and bear), in which the decision to select or not a fund needs to be considered. A risk averse investor will want a portfolio that has low volatility (risk) and will select a portfolio very close to the global minimum variance portfolio, while a risk tolerant investor will ignore volatility and select portfolios with high expected returns.

This problem has been captured within a Gorgias argumentation theory constructed along the following lines. We started by considering the question of when a fund minimally qualifies in the first place to be included or not in a portfolio, thus we identified the primary scenarios that enable our options. So, generally, a fund should not be selected. Only funds with a high return are eligible for selection. This gives the following two scenario-based preferences:
$SP_0(Fund) = \langle S_0 = \{\}; O_0 = \{\neg select(Fund)\}\rangle$
$SP_1(Fund) = \langle S_1 = \{highR(Fund)\}; O_1 = \{select(Fund)\}\rangle$,
whose translation into Gorgias argumentation theory leads to the object level arguments:
$a_0(Fund) : (\{\} \rhd \{\neg select(Fund)\})$
$a_1(Fund) : (\{highR(Fund)\} \rhd \{select(Fund)\})$

Here *highR(Fund)* refers to the fact that the Fund is in the top 30% of the funds with regard to returns.

These primary scenarios were **refined** with the market's characteristics (forecasts) or the investor's attitude. Let's us illustrate some of these:

(1) the bear market context, where a mutual fund should not be selected unless it has low risk:
$SP_{b0}(Fund) = \langle S_{b0} = \{market(bear)\}; O_{b0} = \{\neg select(Fund)\}\rangle$
$SP_{b1}(Fund) = \langle S_{b1} = \{highR(Fund), lowRisk(Fund), lowSTDEV(Fund), market(bear)\};$
$O_{b1} = \{select(Fund)\}\rangle$
(2) the risk tolerant investor doesn't select funds unless they have high returns and high risk (risk also leads to high returns):
$SP_{r0}(Fund) = \langle S_{r0} = \{investor(risk\_tolerant)\}; O_{r0} = \{\neg select(Fund)\}\rangle$
$SP_{r1}(Fund) = \langle S_{r1} = \{highR(Fund), highRisk(Fund), investor(risk\_tolerant)\};$
$O_{r1} = \{select(Fund)\}\rangle$

In generating the corresponding Gorgias argumentation theory it was decided that all specific contexts are to be preferred over their more general contexts. Hence in the general context, a_1 has higher priority over a_2 (default priority) and we have:
$p_1(Fund) : \{\} \rhd (a_1(Fund) > a_0(Fund))$

For the bear market scenario/context we have:
$p_{b0}(Fund) : \{market(bear)\} \rhd (a_0(Fund) > a_1(Fund))$
$p_{b1}(Fund) : \{lowRisk(Fund), lowSTDEV(Fund), market(bear)\} \rhd (a_1(Fund) > a_0(Fund))$
$c_{b0}(Fund) : \{\} \rhd (p_{b0}(Fund) > p_1(Fund))$
$c_{b1}(Fund) : \{\} \rhd (p_{b1}(Fund) > p_{b0}(Fund))$

Similarly, for the risk tolerant investor scenario/context we have:
$p_{r0}(Fund) : \{investor(risk\_tolerant)\} \rhd (a_0(Fund) > a_1(Fund))$
$p_{r1}(Fund) : \{highRisk(Fund), investor(risk\_tolerant)\} \rhd (a_1(Fund) > a_0(Fund))$
$c_{r0}(Fund) : \{\} \rhd (p_{r0}(Fund) > p_1(Fund))$
$c_{r1}(Fund) : \{\} \rhd (p_{r1}(Fund) > p_{r0}(Fund))$

Besides refining the scenarios we considered **combined** scenarios. For example, the bear market context and risk tolerant investor role can be combined, and, in that case, the final portfolio is their union except that the risk tolerant investor now would accept to select high and medium risk funds (instead of only high). Therefore, we have the additional scenario:
$SP_{br1}(Fund) = \langle S_{br1} = \{highR(Fund), mediumRisk(Fund), investor(risk\_tolerant), market(bear)\};$
$O_{br1} = \{select(Fund)\}\rangle.$

Given this we would extend our priority arguments to include:
$p_{br1}(Fund) : \{mediumRisk(Fund), investor(risk\_tolerant), market(bear)\} \rhd (a_1(Fund) > a_0(Fund)))$
$c_{br1\_1}(Fund) : \{\} \rhd (p_{br1}(Fund) > p_{b0}(Fund))$
$c_{br1\_2}(Fund) : \{\} \rhd (p_{br1}(Fund) > p_{r0}(Fund))$

The PORTRAIT tool (as it was named) allowed an investor, or a fund manager, to select the appropriate funds according to the values of financial variables and to the contexts, such as the profile of the investor or the forecasted situation of the market. The results of this work provided evidence that argumentation-based portfolios perform better than the ones based on other, traditional, approaches. The proposed tool was validated using a data set from the Association of Greek Institutional Investors, the Athens Stock Exchange and the Bank of Greece, including daily data of domestic mutual funds, the performance of the market and the return of the three-month Treasury bill respectively, over the period between January 2006 and December 2011. Moreover, the PORTRAIT tool was combined with an evolutionary algorithm for forecasting the market status [47] that had a good performance and helped to have the market context changing dynamically for every investment period.

5. High-level Development of Gorgias Applications

In this section we present a knowledge engineering approach that has emerged out of the experience of the last decade of applying Gorgias to real-life application problems. This new methodological approach allows the high-level development of applications of argumentation where the involvement of the problem domain expert or user can occur within the high-level language of the problem that the expert is familiar with. There is no need for the experts to have any technical knowledge of computational argumentation, in general, or of the specific details of the Gorgias argumentation framework.

Our approach allows a domain expert (e.g. a medical doctor, a lawyer, etc) or a personal user to structure their expertise in a particular domain by using a *table*. In this table the columns represent the

possible options and the rows the scenarios in which the different options are enabled or are preferred. In other words, rows of the table correspond to statements of scenario-based preference.

STEP 1: The expert/user starts filling in the columns of the table with the possible options in the particular application domain. Then s/he starts filling in the rows, by defining for each option O_i (or set of options $\{O_i, ..., O_n\}$) a scenario S_i in which this option (or set of options) hold(s) or are preferred. For each defined scenario the expert puts a mark in the columns containing a preferred option. Several scenarios can be possible for the same option (or set of options) and thus we may have several rows with different scenarios concerning the same option (or set of options). With this the *basic (or initial)* scenarios for the set of possible options are identified.
At the end of this first step possible conflicts between different sets of options might occur. The reason for this can be (i) that in the same scenario there are several different preferred options or (ii) that we have different scenarios that are simultaneously possible as their scenario information can be valid at the same time in an application environment.

STEP 2: The second step of the approach concerns a process of *conflicts analysis* and possible resolution of conflicts. In this step the expert/user detects the conflicting situations and if possible expresses a discrimination among the conflicting options by expressing a preference for a (sub)set of the options involved in the conflict. If no discrimination is possible (or necessary) this would mean that the conflicting options could all be considered as possible solutions under the current scenarios.

STEP 3: When the expert/user decides that s/he can discriminate the options, s/he proceeds to a comparison by possibly introducing additional information in the current scenarios. This gives refinements of the scenarios and the expert/user fills in new rows in the table for each refined scenario. Similarly, the expert/user considers combinations of the current scenarios and if these could possibly hold (i.e they are not inconsistent together or completely disjoint), s/he introduces a new row for such combined scenarios. Then, in these new rows s/he marks "X" in the columns that contain the newly preferred options in these new refined or combined scenarios. This step then implements a first level of conflict analysis and resolution.

STEP 4: Some conflicts might still persist and new ones might appear. The refined scenarios might still be non-deterministic in that they only indicate a subset of preferred options rather than a single preferred option and the new combined scenarios may have new conflicts between the preferred options from each individual scenario. The expert/user examines these conflicts and then repeats the above process of conflict analysis and possible resolution (STEPs 2 and 3). S/he builds further refined scenarios based on the combination of the new current scenarios along with new additional information that s/he considers sufficient to discriminate the conflicting options. New rows are filled in the table with the refined scenarios and marked in the columns containing the preferred options.This gives a new *higher level* of conflict analysis and resolution.

STEP 5: This process can then continue repeating (STEPs 2, 3 and 4) as long as the expert/user is able to build further refined scenarios for resolving conflicts at higher levels.

Note that the approach does not impose any restrictions on the expert in making these statements of scenario-based preferences. The only property that the overall preference relation, captured by these statements, has is that of being irreflexive, which is a simple consequence of the fact that it is not possible to express a preference of an option over itself in the table.

Table 1 shows an example of a table constructed using this approach. At the first step we have the first three rows with the basic scenarios. These scenarios are denoted in the form $S_{\{O_1,...,O_n\}}$. This notation

Table 1
Example of scenario-based preferences on conflicting options

Scenarios \ Options	O_1	O_2	O_3	O_4	O_5	O_6
$S_{\{1,6\}} = \{x_1, x_3, x_6\}$	X					X
$S_{\{2,3,5\}} = \{x_1, x_2, x_8\}$		X	X		X	
$S_{\{4,6\}} = \{x_1, x_3\}$				X		X
$S^1_{\{2\},\{3,5\}} = \{x_1, x_2, x_8, x_9\}$		X				
$S^1_{\{1\},\{6\}} = \{x_1, x_3, x_6\} \cup \{y\}$	X					
$S^1_{\{6\},\{1\}} = \{x_1, x_3, x_6\} \cup \{\neg y\}$						X
$S^1_{\{2,3,5\},\{4,6\}} = \{x_1, x_2, x_3, x_8\}$		X	X		X	
$S^2_{\{4,6\},\{2,3,5\}} = \{x_1, x_2, x_3, x_8\} \cup \{z\}$				X		X

shows the set of options that are enabled (or preferred) in the current scenario. For example, the first three rows present scenarios for individual options or set of options. Note that we may have different scenarios, e.g. $\{x_1, x_3, x_6\}$ and $\{x_1, x_3\}$, where an option, e.g. O_6, is among the preferred ones.

At the next step of conflict analysis and resolution, possible conflicts due to these initial scenarios are identified. For example there is a conflict between options O_1 and O_6 under the scenario $S_{\{1,6\}}$ or between options O_4 and O_6 under the scenario $\{x_1, x_3\}$ which is a subset of the previous scenario $S_{\{1,6\}}$. Once conflicts are detected, the expert can examine whether s/he can discriminate the conflicting options by considering refined scenarios.

Where it is possible to discriminate among conflicts, new rows with refined scenarios are added. These are rows 4 to 7 in the table 1 showing how the expert was able to resolve some conflicts but not all of them. For example, in row 4 the option O_2 is preferred over O_3 and O_5 when the extra scenario information x_9 is added to the scenario $S_{\{2,3,5\}}$). Note that the name, $S^1_{\{2\},\{3,5\}}$ of this refined scenario indicates this preference with the superscript indicating that this is at the first level of conflict analysis. Similarly, in row 5 the option O_1 is preferred over O_6 (see $S^1_{\{1\},\{6\}}$) when the extra information "y" can be verified to hold along with the scenario $\{x_1, x_3, x_6\}$ but O_6 is preferred over O_1 when this does not hold (shown in row 6). This is a clear discriminating situation that does not necessitate a higher level of conflict analysis and resolution as the combination of these two scenarios in rows 5 and 6 leads to an impossible situation.

The table also shows in row 7 that the options O_2, O_3, O_5 are preferred over options O_4,O_6 in the conflict created under the consistent combined scenario $\{x_1, x_2, x_3, x_8\}$ of rows 2 and 3. However at a higher level of conflict analysis of this scenario the options O_4,O_6 are preferred over options O_2, O_3, O_5 when an additional information "z" holds and is integrated in the combined scenario $\{x_1, x_2, x_3, x_8\}$. This is seen in the last row of the table. The superscript 2 in its scenario, $S^2_{\{4,6\},\{2,3,5\}}$, indicates the higher-level step of analysis and resolution of this case.

5.1. Gorgias-B: Tool Support

Gorgias-B is a Java-based tool built on top of *Gorgias* for supporting the development of *Gorgias* argumentation theories[7]. *Gorgias-B* has two important roles in the development of applications of ar-

[7] *Gorgias-B* (http://gorgiasb.tuc.gr) can execute in a computer with the minimum requirements of a Windows or Linux OS, SWI-Prolog version 7.0 or later, and Java version 1.8 or later. The tool employs *Eclipse EMF* (http://eclipse.org) technology (with the *xtext* extension) for managing low level Prolog code and the methodology's high level concepts and transformations from one to the other.

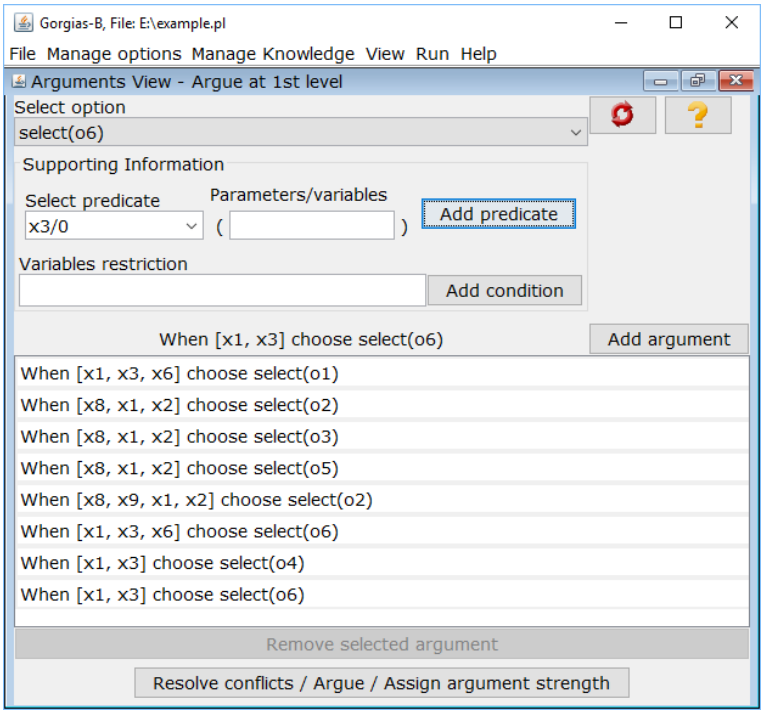

Fig. 1. *Gorgias-B*: The arguments view (step 1 of the methodology)

gumentation. Besides aiding the elicitation of the expert/user knowledge in the form of scenario-based preferences, *Gorgias-B* automatically generates from these a corresponding executable *Gorgias* code. Moreover, *Gorgias-B* allows to execute scenarios so that the user can test the developed argumentation theory, during (or after) the development of the application.

In order to illustrate these roles and the general functionality of *Gorgias-B* we present here how this would be used to capture the scenario-based preferences in the example above, as presented in Table 1. The tool supports the whole process from its start where the different options of an application are declared. In the example of Table 1 the expert/user can then use the "Argument View", shown in Figure 1, to generate object-level arguments for the options. For example, in the figure we see that the user has selected option, O_6, i.e. the predicate *select(o6)*, and has added the predicate conditions *x1* and *x3* and similarly for option O_4, i.e. the predicate *select(o4)*, to form two object-level arguments corresponding to the third row of Table 1. These appear as they are inserted, e.g. "When $[x1, x3]$ choose $select(o6)$", in the main part of the form. Similarly, for each basic scenario (initial rows) in our table supporting a set of options we generate an object-level argument for each option in the set of the row. This appears in Figure 1. Once this has been done for all the basic scenarios, this covers the first step of the methodology.

Continuing with the next steps of conflict analysis and resolution in the approach, the user clicks the button "Resolve conflicts/Argue/Assign argument strength" opening a new dialogue (see Figure 2). This aims to support the development and capture of scenario-based preferences in Steps 2 and 3 of the methodology. *Gorgias-B* identifies scenarios with conflicting options and the user can work on these by selecting (using the controls on the right of the scenario) the option(s) that are preferred (if any) in each conflicting scenario. The user can make these preferences conditional on further contextual information in the scenario under consideration. These preference statements appear in the main window

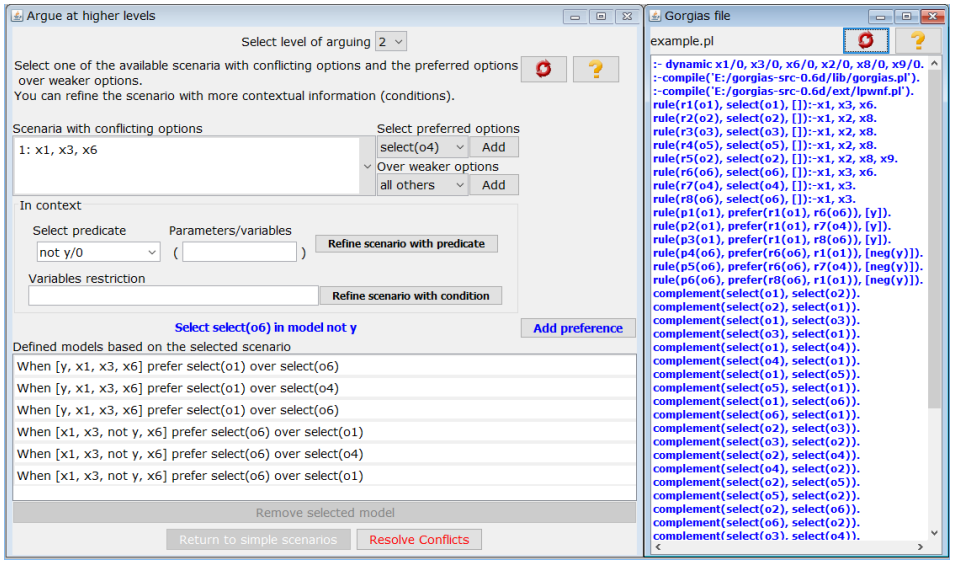

Fig. 2. *Gorgias-B*: Arguing at higher levels (steps 2 and 3 of the methodology)

as binary preferences or priorities between pairs of options. In this way, the tool allows us to enter the scenario-based preferences that are gathered during these two steps. For example, in Figure 2, we see the conflicting scenario $\{x_1, x_3, x_6\}$ where the preference corresponding to row 5 for the refined scenario, $S^1_{\{1\},\{6\}}$ (see Table 1), is inserted, generating three pairwise priorities shown in the main window. Similarly, using the information in row 6 we insert the preference it states by refining the same conflicting scenario, $\{x_1, x_3, x_6\}$, with *"not y"* and selecting O_6 as a preferred option. This again appears as the bottom pairwise priorities in the main window of the Figure 2. The "Gorgias file" view on the right hand side of Figure 2 is optional and allows the user to see the *Gorgias* argumentation theory code that is automatically generated.

Note that the *Gorgias-B* tool would also detect possible conflicts that may not correspond to a row in the corresponding table. For example, it would detect a conflicting scenario of $\{x_1, x_3, x_6\}$ with conflicts between options O_1 and O_4 due to the fact that the basic scenario $S_{\{4,6\}} = \{x_1, x_3\}$ is a subset of the scenario $S_{\{1,6\}}$. The tool, thus, gives us the possibility to resolve this conflict in the same way as in the cases where we already have a second level row in our table. Hence, we can use the tool to give priority of O_1 over O_4 in the refined scenario $\{x_1, x_3, x_6\}$ of the scenario $\{x_1, x_3\}$. Here we also observe that there are two priority arguments of O_1 over O_6 under the scenario $\{x_1, x_3, x_6, y\}$ and two priority arguments of O_6 over O_1 under the scenario $\{x_1, x_3, x_6, \neg y\}$. This is because we had two possible ways for the conflict between these options: one as alternatives in the scenario $\{x_1, x_3, x_6\}$ of row 1 and one from the combination of the two initial scenarios $\{x_1, x_3, x_6\}$ and $\{x_1, x_3\}$. This can be avoided by noticing that the statement in the first row is in fact a refinement of that in the third row, where the additional information of x_6 reverses the preference between O_4 and O_1 in the third row in favour of O_1 now. Such dependencies between scenarios can be discussed with the expert to obtain clarification before proceeding with the transformation into argumentation theories.

By pressing the "Resolve Conflicts" button at the bottom we can move to a next level of conflict analysis corresponding to the higher-level steps of our methodological approach. In the same way, the user adds the scenario-based preferences identified at the higher-level steps and given in the rows of

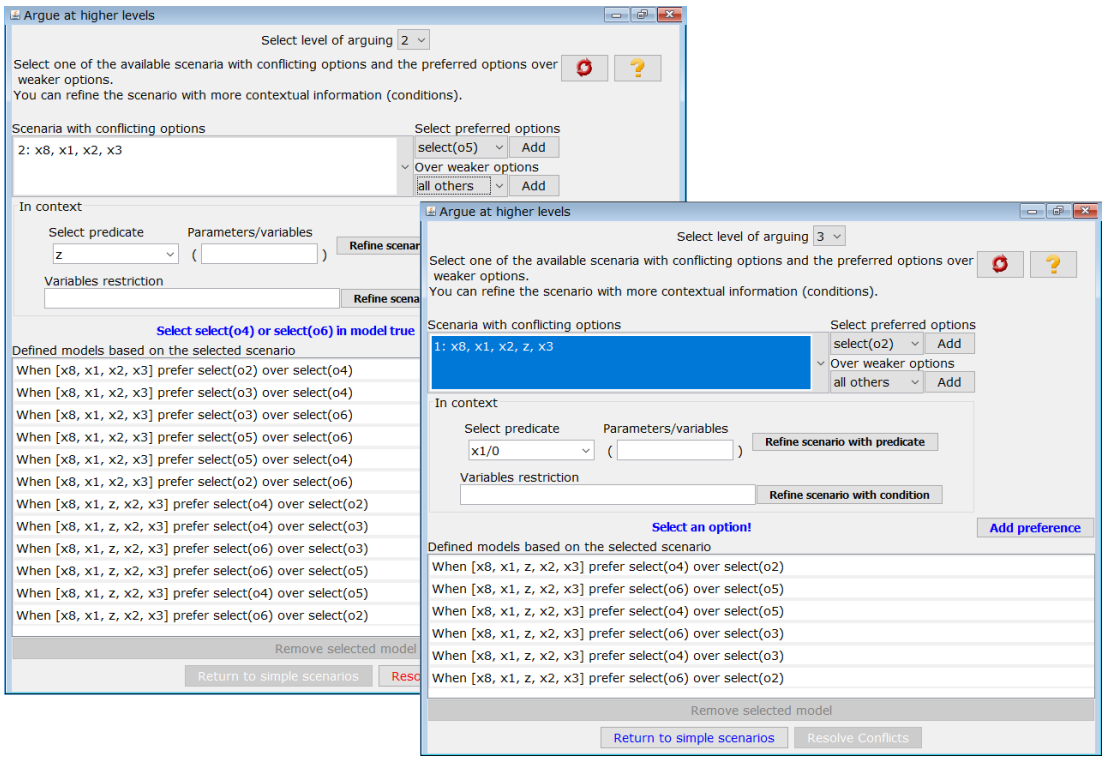

Fig. 3. *Gorgias-B*: Arguing at higher levels (step 4 of the methodology)

our table. Figure 3 shows this for row 7, Table 1, with the scenario $S^1_{\{2,3,5\},\{4,6\}} = \{x_1, x_2, x_3, x_8\}$. We can see that priority statements that correspond to the Cartesian product of the set of preferred options, $\{O_2, O_3, O_5\}$, over the options $\{O_4, O_6\}$ under this scenario are generated.

The row of $S^2_{\{4,6\},\{2,3,5\}}$ in our table indicates, however, that under the more specific scenario $\{x_1, x_2, x_3, x_8, z\}$ the options $\{O_4, O_6\}$ are preferred over the options $\{O_2, O_3, O_5\}$. To capture this, priority statements for options $\{O_4, O_6\}$ over $\{O_2, O_3, O_5\}$ are generated in *Gorgias-B* at the same level, i.e. "2", (as seen in the bottom six statements in the left hand side of Figure 3). These are in conflict with the previous statements of priority of $\{O_2, O_3, O_5\}$, over the options $\{O_4, O_6\}$ but *Gorgias-B* moves to a next level, i.e. level "3", to resolve these conflicts by generating priority statements for options $\{O_4, O_6\}$ under the more specific scenario of $\{x_1, x_2, x_3, x_8, z\}$ (see right hand side of Figure 3). *Gorgias-B* sets automatically these higher-level preferences as the most specific context is automatically assigned preference. Of course, the user can delete unwanted preferences and/or add her own.

The button named “Resolve Conflicts” in the level "2" form of Figure 3 is enabled. This means that in the current scenario *Gorgias-B* has still detected conflicting preferences that need to be resolved at the next level (step 4 of the methodology). As soon as the user clicks the button she arrives at the second screen (level 3) in Figure 3 where such additional resolution statements from the expert can be entered. As mentioned above, *Gorgias-B* shows to the user all combined conflicting scenarios except those for which it has information that they are impossible in practice, e.g. combinations where a condition and its negation are present. The user can then add new preferences to these if such exist or leave them as they are in their conflicting form.

6. Current Development of Applications

In this section we present two real-life applications that are currently under development using the general approach and *Gorgias-B* tool described in the previous section.

6.1. Eye-clinic Support

This application concerns the development of a system that can provide a first level support in (any) eye-clinic by analyzing the symptoms presented by the patient, possibly requesting extra information in order to do so, with the primary aim to identify the urgency of the situation and thus arrange suitably the patient's appointment with the doctor. Sorting the patients and deciding about the urgency of their care according to the severity of their diagnosed disease is an important issue for the ocular emergencies departments in public hospitals (or private clinics). An AI-based diagnosis support system would need to help the receptionist nurse to make a preliminary (i.e. before a doctor's physical examination) diagnosis in order to plan the appointments with the doctors based on the severity of the patient's disease. The receptionist nurse will supply the AI system with observable (or declared by the patient) facts (e.g. symptoms such as red eye, painful eye, etc.) but also the system should be able to prompt the nurse for the need of additional information (e.g. alerting about the need to measure the intra-ocular pressure or asking the patient about past history, etc.). The development of this system was based on a collaboration with a highly qualified ophthalmologist doctor of a public hospital in Paris (France) (see [48]).

The automated diagnosis done by the system is based on the set of known ocular diseases (there are more than 80), which constitute the options in the corresponding decision problem to select the most appropriate ones, given the information gathered from the patient. Options are represented by atoms of the form: $ds(name\_of\_disease, severity)$. For illustrating this application we will consider only four diseases with the set of options thus being (here the severity of the disease is represented by a number whose higher value indicates higher severity):

$$OPTIONS = \{D_1 = ds(viral\_conjunctivitis, 2), D_2 = ds(first\_episode\_uveitis, 2),$$
$$D_3 = ds(recurrent\_uveitis, 3), D_4 = ds(suture\_infection\_after\_surgery, 4)\}.$$

The scenario information consists of observable facts such as the *zone* (e.g. eye), some *symptoms* (e.g. red eye, painful eye) or the *context* (e.g. ocular surgery) but also other hypothetical information such as whether the patient had (or not) a disease antecedent. The scenario-based preferences are provided by the expert ophthalmologist by expressing her natural way of thinking about patient symptoms to arrive at an appropriate diagnosis. Primary scenarios with minimal information allow for several diseases to be initially detected. The expert's analysis of these gives refined or combined scenario-based preferences where the initially preferred diseases are narrowed down or (partly) replaced by other diseases.

In Table 2 we present some of the scenarios that involve the four diseases in our illustrative subset of options. In the description of the following scenarios "z" stands for "zone", "s" for "symptom", "c" for "context", "oc_sur" for "ocular surgery", and "vis_dist" for "visual disturbance". Hence the expert has described three initial scenarios, with their preferred options and, at a second stage, recognizing the existence of a conflict between D_2 and D_3, in the third scenario, the expert has provided the last two extra (refined) scenarios, where additional scenario information can indeed discriminate this conflict.

We can take this table and generate from it a *Gorgias* argumentation theory. To do so we first sort the scenarios involved according to their relative specificity. In the example we have $S_{\{1,2,3,4\}}$ as the most general scenario, then $S_{\{2,3\}}$ and then $S_{\{4\}}$, where the subscript indicates the preferred subset of options. The refinement from $S_{\{1,2,3,4\}}$ to $S_{\{2,3\}}$ narrows down the options but the subsequent refinement of the

Table 2
Example of eye clinic application

Scenarios \ Diseases	D_1	D_2	D_3	D_4
$S_{\{1,2,3,4\}} = \{z(eye), s(red\_eye)\}$	X	X	X	X
$S_{\{4\}} = \{z(eye), s(red\_eye), c(oc\_sur), s(vis\_dist), s(painful\_eye)\}$				X
$S_{\{2,3\}} = \{z(eye), s(red\_eye), s(vis\_dist), s(painful\_eye)\}$		X	X	
$S^1_{\{2\},\{3\}} = \{z(eye), s(red\_eye), s(vis\_dist), s(painful\_eye)\} \cup \{hyp\_info(not\_uv\_atcd)\}$		X		
$S^1_{\{3\},\{2\}} = \{z(eye), s(red\_eye), s(vis\_dist), s(painful\_eye)\} \cup \{hyp\_info(uv\_atcd)\}$			X	

scenario to $S_{\{4\}}$ reinstates D_4 as the preferred option. The last two rows are two different refinements of $S_{\{2,3\}}$ where its preferred options of D_2 and D_3 are focused further. Given this sorting a corresponding *Gorgias* argumentation theory that captures these scenario-based preferences can be directly generated.

Note that these tables do not need to be exhaustive, or to be built completely from the start before analyzing them. The tables can be built incrementally, provided that each time we introduce a conflict analysis step to ascertain if the expert has information that would help resolve, at least partially, possible new conflicts. One of the features of the approach, and this is due generally to argumentation, is that the expert/user may not be able to resolve all conflicts when these first appear. At a later stage and provided that extra knowledge has been acquired, it can be used to contribute to discriminating between existing conflicts. As we saw the *Gorgias-B* tool supports this process of conflict analysis and possible resolution.

Our methodology has allowed the structuring of the high level expertise of an ophthalmologist in several tables in a systematic and flexible way with respect to the complexity and the amount of the knowledge to be extracted. The use of tables provides a compact representation of the expert knowledge, but, more importantly, gives the opportunity to our expert to analyze several times new scenarios based on the combination of basic scenarios when she had to compare and discriminate conflicting diseases through scenario-based preferences. A prototype is under development and we already have implemented more than 3000 rules representing object and priority arguments. A commercial product will be developed from the prototype in the near future to be initially deployed in the collaborating hospital.

6.2. Data Access and Sharing

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 data sharing agreement (DSA). The works in [39, 49] apply an argumentation approach within the Gorgias framework for data access and sharing with a specific application in the health sector. In [50], a web application, called *MEDICA* [8], was developed for the purpose of determining the level of access to patients' data records, according to an EU country's entire 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. 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 or priority level) arguments that support the decision reached. The system also supports requests from users wanting to gain higher level of access by indicating the extra information that needs to hold, if such access is possible.

Currently, we are developing the *MEDICA* application further with the help of a focus group of specialists to assess the applicability of the system for usage in hospitals and health centers. We are

[8] http://medica.cs.ucy.ac.cy

working closely with a government advisory team for IT systems for the national health service. This team is evaluating the system from their own IT perspective and then will proceed for an evaluation through a pilot trial at appropriate medical centers.

We are also working on applications requiring the merging of several diverse policies from independent organizations or people. The main challenge in this is to manage conflicts between policies where we need to understand how one policy's preference takes precedence over the others. Such is the situation in the data sharing landscape (see e.g. [39, 51]). Data is generated in independent and distributed nodes and access to it by different stakeholders is governed by a diverse set of policies. Stakeholders are independent entities all with their own policies and goals. Typically, there are several cases where their policies will conflict and in that case an arbitrator is needed to handle the conflicts. We can use our approach and Gorgias-B to help manage these conflicts and help to resolve them.

We do this by building an arbitrator *meta policy* over the individual policies. From an appropriate authority or through consultation between the stakeholders we build an application to capture such a meta policy. Following our approach we formulate tables of scenario-based preferences as statements of preferences between the individual policies which take the role of the options for the meta policy. For example, in a medical data access application where we have several stakeholders such as the Patient, Legislation, Hospital and Emergency Organizations (e.g. Fire Service) such a meta policy can be built from statements of the form: the personal and legislation policies are preferred over others and among the two the personal policy is preferred. However, if the person is a victim in the scene of an accident, the fire service policy is preferred over the personal one. In a situation when the owner of the data is hospitalized the hospital policy is preferred.

This approach allows the arbitrator to be agnostic with regard to the reasons why a policy allows or denies access and at the same time be capable to arbitrate on such a matter. If, for example, the requester is a treating doctor in a hospital the above arbitration policy gives preference to the hospital policy. The hospital policy could deny access to the doctor, e.g. when the hospital policy allows access only for patients in intensive care. In the case, however, that the treating doctor is not in a hospital, then the patient's policy presides and this could grant access to a treating doctor.

Apart from applications on medical data shared agreements we are also working on applying such meta policy arbitration to smart green buildings [52]. In this domain, there are, again, different stakeholders whose policies can be conflicting. In a corporate environment, for example, there is the policy of the building manager that prefers to limit consumption of energy and restrict the use of air-conditioners, the policy of a system administrator that requires heavy use of the air-conditioners for the computer room, the policy of the company management and other employees with their own policies at their offices. Again, our approach can provide the means to arbitrate conflicts and merge these policies.

7. Conclusion

We have presented an approach for the high-level development of a (class) of applications of argumentation that has emerged out of the experience of the last decade of applying the argumentation framework of *Gorgias* to real-life application problems. The proposed approach allows the modular development of real-life applications where requirements are accommodated incrementally in their high-level form, through the systematic analysis of scenario-based conflicts and an automatic translation of scenario-based preferences to corresponding *Gorgias* argumentation theories and code. The approach is sufficiently general to allow the development of applications in any, of the many existing argumentation frameworks (e.g. [8, 53, 54]) that support conditional and hierarchical forms of preference.

The approach can be improved in several ways. An important development of the interface tool of *Gorgias-B* is the fully automated extraction of the Gorgias argumentation theory directly from the tables of scenario-based preferences so that while these tables are filled the argumentation code can be produced and tested. Given the current investor interest that we have in this we hope to develop a new interface tool for professional use in software companies.

We can also integrate within the development approach a process of learning, where scenario-based preferences can be extracted without the need for their explicit stipulation by the domain expert/user. For example, by including a feedback process during the operation of the system we can learn from experience new or refined preferences. This feedback can be linked to the provision of explanations that argumentation systems can offer for their operation and decisions. We are thus working to extend the explanation facility of the interface tools to give these in a natural form familiar to the users and to engage in dialogues with the users.

We are also experimenting with the development of Cognitive Assistants in various areas of applications, e.g. personal, tourist, calendar, shopping and social media assistants. This class of applications relies on user guidelines expressed in natural language. A suitable extension of the *Gorgias* system, called *Gorgias-NL* [55] is currently under development where scenario-based preferences would be extracted directly from dialogues with the user in structured forms of natural language.

References

[1] H. Mercier and D. Sperber, Why do humans reason? Arguments for an argumentative theory, *Behavioral and brain sciences* **34**(2) (2011), 57–74.

[2] F. Toni and P. Torroni, Theorie and Applications of Formal Argumentation: First International Workshop, TAFA 2011. Barcelona, Spain, July 16-17, Revised Selected Papers, S. Modgil, N. Oren and F. Toni, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 249–262, Chap. Bottom-Up Argumentation. ISBN 978-3-642-29184-5.

[3] J. Lawrence, J. Park, K. Budzynska, C. Cardie, B. Konat and C. Reed, Using Argumentative Structure to Interpret Debates in Online Deliberative Democracy and eRulemaking, *ACM Trans. Internet Techn.* **17**(3) (2017), 25–12522.

[4] I. Gurevych, M. Lippi and P. Torroni, Argumentation in Social Media, *ACM Trans. Internet Techn.* **17**(3) (2017), 23–1232.

[5] T.J.M. Bench-Capon, H. Prakken and G. Sartor, Argumentation in Legal Reasoning, in: *Argumentation in Artificial Intelligence*, 2009, pp. 363–382.

[6] H. Prakken and G. Sartor, Law and Logic: a Review from an Argumentation Perspective, *Artificial Intelligence* **227** (2015), 214–245.

[7] D. Gaertner and Francesca, Computing Arguments and Attacks in Assumption-Based Argumentation, *IEEE Intelligent Systems* **22**(6) (2007), 24–33. doi:10.1109/MIS.2007.105.

[8] A.J. García and G.R. Simari, Defeasible Logic Programming: An Argumentative Approach, *Theory and Practice of Logic Programming* **4**(2) (2004), 95–138, ISSN 1471-0684. doi:10.1017/S1471068403001674.

[9] M. Snaith and C. Reed, TOAST: Online ASPIC$^+$ implementation, in: *Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012*, 2012, pp. 509–510. doi:10.3233/978-1-61499-111-3-509.

[10] A. Kakas and P. Moraitis, Argumentation Based Decision Making for Autonomous Agents, in: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, ACM, New York, NY, USA, 2003, pp. 883–890. ISBN 1-58113-683-8.

[11] H. Lindgren and P. Eklund, Differential diagnosis of dementia in an argumentation framework, *Journal of Intelligent and Fuzzy Systems* **17**(4) (2006), 387–394.

[12] R. Craven, F. Toni, C. Cadar, A. Hadad and M. Williams, Efficient Argumentation for Medical Decision-Making, in: *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*, 2012.

[13] C. Dong and N. Dulay, Argumentation-based Fault Diagnosis for Home Networks, in: *Proceedings of the 2nd ACM SIGCOMM Workshop on Home Networks*, HomeNets '11, ACM, New York, NY, USA, 2011, pp. 37–42. ISBN 978-1-4503-0798-7. doi:10.1145/2018567.2018576.

[14] T. Bench-Capon, H. Prakken and G. Sartor, Argumentation in Legal Reasoning, in: *Argumentation in Artificial Intelligence*, G. Simari and I. Rahwan, eds, Springer US, Boston, MA, 2009, pp. 363–382.

[15] A. Mocanu, X. Fan, F. Toni, M. Williams and J. Chen, Combinations of Intelligent Methods and Applications: Proceedings of the 4th International Workshop, CIMA 2014, Limassol, Cyprus, November 2014 (at ICTAI 2014), I. Hatzilygeroudis, V. Palade and J. Prentzas, eds, Springer International Publishing, Cham, 2016, pp. 97–115, Chap. Online Argumentation-Based Platform for Recommending Medical Literature. ISBN 978-3-319-26860-6. doi:10.1007/978-3-319-26860-6_6.

[16] S. Huang and C. Lin, The search for potentially interesting products in an e-marketplace: An agent-to-agent argumentation approach, *Expert Systems with Applications* **37**(6) (2010), 4468–4478, ISSN 0957-4174. doi:10.1016/j.eswa.2009.12.064.

[17] G. Wang, T.N. Wong and X.H. Wang, A Negotiation Protocol to Support Agent Argumentation and Ontology Interoperability in MAS-Based Virtual Enterprises, in: *Seventh International Conference on Information Technology: New Generations, ITNG 2010, Las Vegas, Nevada, USA, 12-14 April 2010*, 2010, pp. 448–453. doi:10.1109/ITNG.2010.39.

[18] K. Pashaei, F. Taghiyareh and K. Badie, A Negotiation-Based Genetic Framework for Multi-Agent Credit Assignment, in: *Multiagent System Technologies - 12th German Conference, MATES 2014, Stuttgart, Germany, September 23-25, 2014. Proceedings*, 2014, pp. 72–89. doi:10.1007/978-3-319-11584-9_6.

[19] M. Aulinas, P. Tolchinsky, C. Turon, M. Poch and U. Cortés, Argumentation-based framework for industrial wastewater discharges management, *Engineering Applications of Artificial Intelligence* **25**(2) (2012), 317–325, Special Section: Local Search Algorithms for Real-World Scheduling and Planning, ISSN 0952-1976. doi:10.1016/j.engappai.2011.09.016.

[20] H.K.H. Chow, W. Siu, C. Chan and H.C.B. Chan, An argumentation-oriented multi-agent system for automating the freight planning process, *Expert Systems with Applications* **40**(10) (2013), 3858–3871, ISSN 0957-4174.

[21] N.R. Velaga, N.D. Rotstein, N. Oren, J.D. Nelson, T.J. Norman and S. Wright, Development of an integrated flexible transport systems platform for rural areas using argumentation theory, *Research in Transportation Business and Management* **3** (2012), 62–70, Flexible Transport Services, ISSN 2210-5395. doi:http://dx.doi.org/10.1016/j.rtbm.2012.05.001.

[22] W. Zhang, Y. Liang, S. Ji and Q. Tian, Argumentation agent based fire emergency rescue project making, in: *Robotics and Applications (ISRA), 2012 IEEE Symposium on*, 2012, pp. 892–895. doi:10.1109/ISRA.2012.6219335.

[23] A. Hunter and M. Williams, Aggregation of Clinical Evidence Using Argumentation: A Tutorial Introduction, in: *Foundations of Biomedical Knowledge Representation - Methods and Applications*, A. Hommersom and P.J.F. Lucas, eds, Springer International Publishing, 2015, pp. 317–337. doi:10.1007/978-3-319-28007-3_20.

[24] M. Makriyiannis, T. Lung, R. Craven, F. Toni and J. Kelly, Combinations of Intelligent Methods and Applications: Proc. of the 4th International Workshop, CIMA 2014, Limassol, Cyprus, November 2014 (at ICTAI 2014), I. Hatzilygeroudis, V. Palade and J. Prentzas, eds, Springer Int. Publishing, Cham, 2016, pp. 79–95, Chap. Smarter Electricity and Argumentation Theory. ISBN 978-3-319-26860-6. doi:10.1007/978-3-319-26860-6_5.

[25] J. Fox, D. Glasspool, V. Patkar, M. Austin, L. Black, M. South, D. Robertson and C. Vincent, Delivering clinical decision support services: There is nothing as practical as a good theory, *Journal of Biomedical Informatics* **43**(5) (2010), 831–843.

[26] A.C. Kakas, P. Mancarella and P.M. Dung, The Acceptability Semantics for Logic Programs, in: *Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy, June 13-18, 1994*, 1994, pp. 504–519.

[27] Y. Dimopoulos and A.C. Kakas, Logic Programming without Negation as Failure, in: *Logic Programming, Proceedings of the 1995 International Symposium, Portland, Oregon, USA, December 4-7, 1995*, 1995, pp. 369–383.

[28] Y. Dimopoulos and A.C. Kakas, Logic Programming without Negation as Failure, in: *Logic Programming, Proceedings of the 1995 International Symposium, Portland, Oregon, USA, December 4-7, 1995*, 1995, pp. 369–383.

[29] A.C. Kakas, P. Mancarella, F. Sadri, K. Stathis and F. Toni, Computational Logic Foundations of KGP Agents, *Journal of Artificial Intelligence Research* **33** (2008), 285–348.

[30] A. Kakas, N. Maudet and P. Moraitis, Modular Representation of Agent Interaction Rules through Argumentation, *Autonomous Agents and Multi-Agent Systems* **11**(2) (2005), 189–206, ISSN 1573-7454. doi:10.1007/s10458-005-2176-4.

[31] A.C. Kakas and P. Moraitis, Adaptive agent negotiation via argumentation, in: *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, 2006, pp. 384–391.

[32] P. Moraitis and N.I. Spanoudakis, Argumentation-Based Agent Interaction in an Ambient-Intelligence Context, *IEEE Intelligent Systems* **22**(6) (2007), 84–93. doi:10.1109/MIS.2007.101.

[33] A.C. Kakas, R. Miller and F. Toni, An Argumentation Framework of Reasoning about Actions and Change, in: *Logic Programming and Nonmonotonic Reasoning, 5th International Conference, LPNMR'99, El Paso, Texas, USA, December 2-4, 1999, Proceedings*, Lecture Notes in Computer Science, Vol. 1730, Springer, 1999, pp. 78–91. ISBN 3-540-66749-0.

[34] I.A. Letia and M. Acalovschi, Achieving Competence by Argumentation on Rules for Roles, in: *Engineering Societies in the Agents World V: 5th International Workshop, ESAW 2004, Toulouse, France, October 20-22, 2004. Revised Selected and Invited Papers*, M.-P. Gleizes, A. Omicini and F. Zambonelli, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 45–59. ISBN 978-3-540-31887-3. doi:10.1007/11423355_4.

[35] J. Marcais, N. Spanoudakis and P. Moraitis, Using Argumentation for Ambient Assisted Living, in: *Artificial Intelligence Applications and Innovations: 12th INNS EANN-SIG International Conference, EANN 2011 and 7th IFIP WG 12.5 International Conference, AIAI 2011, Corfu, Greece, September 15-18, 2011, Proceedings , Part II*, L. Iliadis, I. Maglogiannis and H. Papadopoulos, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 410–419. ISBN 978-3-642-23960-1. doi:10.1007/978-3-642-23960-1_48.

[36] N. Spanoudakis and P. Moraitis, Engineering Ambient Intelligence Systems Using Agent Technology, *IEEE Intelligent Systems* **30**(3) (2015), ISSN 15411672. doi:10.1109/MIS.2015.3.
[37] Y. Benazzouz and D. Boyle, Argumentation-Based Conflict Resolution in Pervasive Services, in: *Negotiation and Argumentation in Multi-agent Systems: Fundamentals, Theories, Systems and Applications*, F. Lopes and H. Coelho, eds, Bentham Science Publishers, 2014, pp. 399–419. ISBN 978-1-60805-825-9.
[38] N.I. Spanoudakis, A.C. Kakas and P. Moraitis, Conflicts Resolution with the SoDA Methodology, in: *Conflict Resolution in Decision Making: Second International Workshop, COREDEMA 2016, The Hague, The Netherlands, August 29-30, 2016, Revised Selected Papers*, R. Aydoğan, T. Baarslag, E. Gerding, C.M. Jonker, V. Julian and V. Sanchez-Anguix, eds, Springer International Publishing, Cham, 2017, pp. 82–99. ISBN 978-3-319-57285-7.
[39] E. Karafili and E.C. Lupu, Enabling Data Sharing in Contextual Environments: Policy Representation and Analysis, in: *Proceedings of the 22Nd ACM on Symposium on Access Control Models and Technologies*, SACMAT '17 Abstracts, ACM, New York, NY, USA, 2017, pp. 231–238. ISBN 978-1-4503-4702-0. doi:10.1145/3078861.3078876.
[40] A.K. Bandara, A.C. Kakas, E.C. Lupu and A. Russo, Using argumentation logic for firewall configuration management, in: *2009 IFIP/IEEE International Symposium on Integrated Network Management*, 2009, pp. 180–187, ISSN 1573-0077.
[41] T.E. Uribe and S. Cheung, Automatic Analysis of Firewall and Network Intrusion Detection System Configurations, in: *Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering*, FMSE '04, ACM, New York, NY, USA, 2004, pp. 66–74. ISBN 1-58113-971-3. doi:10.1145/1029133.1029143.
[42] A. Applebaum, Z. Li, K. Levitt, S. Parsons, J. Rowe and E.I. Sklar, Firewall configuration: An application of multiagent metalevel argumentation, *Argument & Computation* **7**(2–3) (2016), 201–221.
[43] N. Spanoudakis and P. Moraitis, Engineering an agent-based system for product pricing automation, *Engineering Intelligent Systems for Electrical Engineering and Communications* **17**(2–3) (2009), 139–151, ISSN 1472-8915.
[44] F. Cloppet, P. Moraitis and N. Vincent, An Agent-Based System for Printed/Handwritten Text Discrimination, in: *PRIMA 2017: Principles and Practice of Multi-Agent Systems: 20th International Conference, Nice, France, October 30 – November 3*, B. An, A. Bazzan, J. Leite, S. Villata and L. van der Torre, eds, Springer International Publishing, Cham, 2017, pp. 180–197. ISBN 978-3-319-69131-2. doi:10.1007/978-3-319-69131-2_11.
[45] K. Pendaraki and N. Spanoudakis, Portfolio performance and risk-based assessment of the PORTRAIT tool, *Operational Research* **15**(3) (2015), 359–378, ISSN 1866-1505. doi:10.1007/s12351-014-0162-9.
[46] C. Zopounidis and M. Doumpos, Multicriteria decision systems for financial problems, *TOP* **21**(2) (2013), 241–261.
[47] N. Spanoudakis, K. Pendaraki and G. Beligiannis, Portfolio construction using argumentation and hybrid evolutionary forecasting algorithms, *International Journal of Hybrid Intelligent Systems* **6**(4) (2009), 231–243, ISSN 1448-5869.
[48] A. Pison, Développement d'un outil d'aide au triage des patients aux urgences ophtalmologiques par l'infirmière d'accueil à l'aide du système d'argumentation Gorgias-B, Technical Report, LIPADE, Paris Descartes University, 2017.
[49] E. Karafili, K. Spanaki and E.C. Lupu, An argumentation reasoning approach for data processing, *Computers in Industry* **94**(Supplement C) (2018), 52–61, ISSN 0166-3615. doi:https://doi.org/10.1016/j.compind.2017.09.002.
[50] N.I. Spanoudakis, E. Constantinou, A. Koumi and A.C. Kakas, Modeling Data Access Legislation with Gorgias, in: *Advances in Artificial Intelligence: From Theory to Practice: 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, Arras, France, June 27-30, Part II*, S. Benferhat, K. Tabia and M. Ali, eds, Springer International Publishing, Cham, 2017, pp. 317–327. ISBN 978-3-319-60045-1.
[51] F. Martinelli, I. Matteucci, M. Petrocchi and L. Wiegand, A Formal Support for Collaborative Data Sharing, in: *Multidisciplinary Research and Practice for Information Systems: IFIP WG 8.4, 8.9/TC 5 International Cross-Domain Conference and Workshop on Availability, Reliability, and Security, CD-ARES 2012, Prague, Czech Republic, August 20-24*, G. Quirchmayr, J. Basl, I. You, L. Xu and E. Weippl, eds, Springer Berlin Heidelberg, 2012, pp. 547–561. ISBN 978-3-642-32498-7.
[52] T.G. Stavropoulos, E. Kontopoulos, N. Bassiliades, J. Argyriou, A. Bikakis, D. Vrakas and I. Vlahavas, Rule-based approaches for energy savings in an ambient intelligence environment, *Pervasive and Mobile Computing* **19**(Supplement C) (2015), 1–23, ISSN 1574-1192. doi:https://doi.org/10.1016/j.pmcj.2014.05.001.
[53] S. Modgil, Hierarchical Argumentation, in: *Logics in Artificial Intelligence, 10th European Conference, JELIA 2006, Liverpool, UK, September 13-15, 2006, Proceedings*, 2006, pp. 319–332. doi:10.1007/11853886_27.
[54] S. Modgil and H. Prakken, The *ASPIC*$^+$ framework for structured argumentation: a tutorial, *Argument & Computation* **5**(1) (2014), 31–62. doi:10.1080/19462166.2013.869766.
[55] T. Mitsikas, N. Spanoudakis, P. Stefaneas and A. Kakas, From Natural Language to Argumentation and Cognitive Systems, in: *Proceedings of the 13th International Symposium on Commonsense Reasoning*, London, UK, 2017.