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 with the *Gorgias* preference-based structured argumentation framework of *Logic Programming with Priorities*. 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 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 technical details of argumentation. A new tool, called *Gorgias-B*, supports this high-level development of applications and automatically generates the underlying argumentation theory in *Gorgias* code. The paper also reports on ongoing real-life applications in two domains: an eye-clinic assistant for first level support, and systems for patient data access and data sharing agreements according to the relevant legislation and contracts or policies of the several stakeholders involved. The proposed approach is quite general for this type of applications of argumentation and, thus, it can be used with any preference-based argumentation framework.

Keywords: Artificial Intelligence, Argumentation, Applications

1. Introduction

Over the last two decades there has been an ever increasing interest in applying argumentation to analyze and solve practical problems in the area of Artificial Intelligence (AI). The natural link of argumentation to human reasoning (see, e.g., [1]) makes Computational Argumentation an appropriate framework for applications in AI. This suitability of argumentation applies to problems with strict specifications, requiring expert (e.g. scientific) reasoning, as well as to problems of human-like AI, where the reasoning required is closer to that of common sense used by people when they tackle their everyday tasks. We, therefore, see applications of argumentation over a wide spectrum of problems, ranging from applications focusing on the analysis of debates (e.g. debates on on-line social interaction settings [2–4]) to problems where a more formal structure of argumentation needs to be captured computationally (e.g. in automating legislation [5, 6]). There are also several systems of argumentation, such as CaSAPI [7], DeLP [8], TOAST [9] and Gorgias, [10] that provide tool support for studying various aspects of computational argumentation and its application.

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

A wide class of applications of argumentation concerns decision problems in dynamic and incomplete, or uncertain, environments. These include problems of diagnosis, e.g. in the medical domain [11, 12] or in other domains such as home networks [13], legal problems [14], or recommendation problems, e.g. medical treatment recommendation systems [15]. More generally, some practical works in the category of "decision making" applications include e-commerce applications [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 works 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 goal 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.

More generally, in today's era of *human-like AI* and *Cognitive Computing*, there are three main challenges to developing applications, which are particularly well served by an approach based on argumentation and which our approach aims to tackle. Firstly, the acquisition and elicitation of the application requirements needs to be carried out at a high-level 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 their development. Finally, systems need to be accountable, with their decisions or recommendations being *explainable* to people, a feature that is now required by law in the European Union. In effect, the systems need to be able to argue, in a natural human-like way, about the suitability of their operation with respect to the high-level requirements or guidelines provided by the application owners.

Our approach to developing such applications of argumentation 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] implementation. These include applications in medical support systems, network security, business computing, ambient intelligence, pervasive services and, recently, cognitive personal assistants.

This approach is based on the consideration of application scenarios and the provision, by the domain expert, or the application owner and/or user, of statements of preference on the possible solutions within these scenarios. The expert or owner is guided to analyze the conflicts in the application scenarios or combinations of these and to consider possible refinements of the scenarios that could resolve or mitigate the conflict. The important characteristic of this approach is that it can be carried out at a high level familiar to the domain expert or application owner. It abstracts away from the technical details of argumentation through the possibility of automatically transforming the high-level specification into an argumentation theory and executable [10, 26, 27] *Gorgias* code. As such, our approach aims to address, at least partly, the three challenges for today's AI systems cataloged above.

The paper briefly reviews the preference-based argumentation framework of *Gorgias* together with various real-life applications of argumentation based on this framework. The methodological approach

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

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

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 tool, called *Gorgias-B*, employing a novel human-machine interface, supports the proposed approach and provides an automatic translation of the high-level scenario-based preference specification into a corresponding argumentation theory and *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 case and thus helping in prioritizing patient appointments. Similarly, we report on the current development of data access and sharing applications for patient medical records where multiple policy requirements from legislation and other stakeholders need to be integrated together to regulate the access to information.

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 so far with *Gorgias*. Subsequently, in section five, we present the general methodological approach for applications of argumentation, along with the *Gorgias-B* tool developed for supporting the proposed approach. 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 using argumentation are in essence *decision problems*. Argumentation is particularly suitable when the application environment is incomplete and dynamic. The incompleteness and volatility of information typically mean that there is insufficient information to have a strict decision policy and, hence, the flexibility of argumentation provides a way to account for and manage the possible alternative decisions that can be taken.

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: This is the 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 sensitive data. 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 recommend.

SCENARIO INFORMATION: A set of relations that are used to (partially) describe the possible states of the application environment called *scenarios*. These relations 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 process to solve particular instances of the problem. This information can be sensory information readily available 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 separates the sensory information into two types: *directly observable* and *hypothetical* with the latter requiring a decision to actively seek it.

SCENARIO-Based PREFERENCES: A set of tuples $\langle S\,;O\rangle$ of (partial) scenarios, S, together with a corresponding preferred subset, O, of options in the scenario S. 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 specifying decision problems allows the application domain experts to formulate and describe their application at a high and non-technical level solely within their familiar application language. The domain expert does not need to be aware of the technical details of the underlying argumentation framework in which the problem will eventually be expressed. Hence, the Scenario-based Preferences (SP), capture 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) provides 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 personal 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, such as for example, the preference for good quality food or the avoidance of red meat, in the context of building on-line shopping personal assistants.

2.1. Scenarios and Scenario-based Preferences

In expressing the scenario and scenario-based preference specification of an application problem there are two important notions that help us to relate and organize these requirements. Firstly, given a scenario S, we can expand this with further, non-empty, scenario information C, to obtain a new scenario S', $S' = S \cup C$. S' will be called a **refinement** of S, as S' provides a more refined description than that of S. Successive scenario refinements result in a **hierarchy of scenarios** starting from an initial scenario S. We will denote such a hierarchy by S^k $(k = 1, 2, ..., n)$ where $S^1 = S$ and $S^k = S^{k-1} \cup C^k$ $(k > 0)$, where C^k is a non-empty set of scenario information disjoint from S^{k-1}, referred to as the **context** at the k^{th} level of the hierarchy. Scenarios can be identified by a subscript, e.g. S_α, where α can be any symbol, normally one that abbreviates the scenario description. Given two (initial) scenarios S_α and S_β, their **combination** is a new scenario obtained simply by their set union[3]. We will denote a combination by $S_{\alpha,\beta}$, i.e. $S_{\alpha,\beta} = S_\alpha \cup S_\beta$.

Refinements and combinations of scenarios have an effect on the corresponding scenario-based preferences that refer to the scenarios involved. Given two scenario-based preferences $SP = (S; O)$ and $SP' = (S'; O')$ such that S' is a refinement of S, i.e. $S' \supset S$, we say that SP' is a **refinement of** SP. When in addition $O' \subseteq O$, we will say that SP' is a **focusing of** SP, as SP' focuses, or sharpens, the preference statement of SP. Similarly, a hierarchy of scenarios, S^k, induces a **hierarchy** $SP^k = (S^k; O^k)$ on the set of scenario-based preferences. When SP^k is a focusing of SP^{k-1} for any $k > 1$, i.e. when $O \supseteq O^1 \supseteq O^2 \supseteq ... \supseteq O^n$ we say that SP^k is a **focusing hierarchy of scenario-based preferences**.

[3]Clearly, the combination of two (different) scenarios also forms a refinement of each scenario by the other where the non-common part of the scenarios forms the context of the refinement.

Note that when a refinement, $SP' = (S'; O')$, of a given scenario-based preference, $SP = (S; O)$, is not also a focusing of SP, the set O' of preferred options in SP' can be completely different: O' can contain options not in O and can possibly be disjoint from O. This is an indication of a **change of context** in the preference statements when we move from S to S'. A typical case of this, is the change from a default context in general situations described by S to exceptional or specialized contexts when some additional specific information is considered in a refined scenario S'. Similarly, when we are considering the combination of two scenarios the preferred options in the combined scenario can be any subset of options that is independent from the sets of the preferred options in the individual scenarios. Hence this information of the preferred options in the combined scenario must be obtained independently from the same source (application expert/owner/user) that provides the scenario-based preferences.

Finally, we note that when we refine or combine scenarios we need to know if the resulting scenarios are **plausible**, i.e. they indeed describe possible cases of the application environment, or in other words, that they are not inconsistent. Again this information of plausibility of scenarios needs to be provided independently by the source of the application requirements.

2.2. An illustrative Example

In order to illustrate the 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 shopping personal assistant. The set of options in this problem is to buy, or not, 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 application problem is to decide which *buy* options 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. The user has informed us that all these options are enabled under the minimal scenario information of "main shopping" for the week (denoted here by *main_shopping*). This can be captured by a **basic or initial** scenario-based preference statement as follows[4]:
$SP^1_m = \langle S^1_m = \{main\_shopping\}; O^1_m = \{buy(lamb), buy(pork), buy(chicken), buy(fish)\}\rangle$

We will say that these options are **enabled** or **available** in the basic scenario S^1_m. The user can then express preferences on these enabled options depending on different further scenario information that is indeed sufficient for a meaningful preference to be expressed. For example, the user can prefer the cheaper options, expressing the preference for the typically cheaper foods of pork and chicken:
$SP^2_{m,c} = \langle S^2_{m,c} = S^1_m \cup \{cheap(pork), cheap(chicken)\}; O^2_{m,c} = \{buy(pork), buy(chicken)\}\rangle$

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. These additional preferences may be captured in further scenarios, which are **refinements** of the scenario $S^2_{m,c}$, in the same way that $S^2_{m,c}$ is a refinement of the initial scenario S^1_m:
$SP^3_{m,c,w} = \langle S^3_{m,c,w} = S^2_{m,c} \cup \{winter\}; O^3_{m,c,w} = \{buy(pork)\}\rangle$
$SP^3_{m,c,s} = \langle S^3_{m,c,s} = S^2_{m,c} \cup \{summer\}; O^3_{m,c,s} = \{buy(chicken)\}\rangle$

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 example, the user may have another preference amongst the cheaper options, e.g. for the locally produced foods. Hence we can have a new refined scenario-based preference of:
$SP^3_{m,c,l} = \langle S^3_{m,c,l} = S^2_{m,c} \cup \{local(chicken)\}; O^3_{m,c,l} = \{buy(chicken)\}\rangle$

[4] As defined above, the general notation that we will use to denote scenarios (and correspondingly scenario-based preferences) is S^x_y, where y is an optional abbreviated scenario description symbol and x stands for the refinement level.

when the cheap chicken is also produced locally.

When we have different scenarios whose conditions can hold together in the application then we are led to consider **combinations** of scenarios, i.e. a new scenario made up of the union of the two scenarios. The user typically considers the preferred options amongst the union of the preferred options in the two scenarios. This occurs, for example, when it is possible that different refinements of a scenario occur together, e.g. when in our example both the refinements of *winter* and $local(chicken)$ hold together:
$SP^4_{m,c,w,l} = \langle S^4_{m,c,w,l} = S^3_{m,c,l} \cup S^3_{m,c,w} = S^2_{m,c} \cup \{winter\} \cup \{local(chicken)\}; O^4_{m,c,w,l} = \{buy(chicken)\}\rangle$,
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 stemming from the scenario condition of "cheap 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, $SP^4_{m,c,w,l}$ 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^1_m:
$SP^2_{m,l} = \langle S^2_{m,l} = S^1_m \cup \{local(chicken), local(lamb)\}; O^2_{m,l} = \{buy(chicken), buy(lamb)\}\rangle$
when only chicken and lamb are locally produced. We would then continue by considering refinements or combinations of this scenario, e.g., its combination with $SP^2_{m,c}$ or its refinement with the context of winter or summer.

In certain applications, it is possible for the list of preferred options in the new refined or combined scenarios, to contain options that are outside the (union of) options in the "parent" scenario-based preferences. For example, the user prefers lamb for special occasions and chicken when s/he is not feeling well, but when both these hold (i.e. in the combined scenario of a special occasion when s/he is not feeling well) s/he prefers fish. Finally, we note that we can also have "local" preference statements between an option and its negation, e.g, the user may express the preference not to buy fish if it is farmed:
$SP^1_{ff} = \langle S^1_{ff} = \{farm(fish)\}; O^1_{ff} = \{not(buy(fish))\}\rangle$
These are "vertical" preferences that apply locally only to the particular option involved.

Some of the scenario information may not be readily available to the shopping assistant at the time of operation, e.g. in our example the user may have expressed a preference for buying lamb or fish when the week contains a special occasion but the information of *special_occasion* may not always be directly available to the assistant. We can then designate such conditions as **hypothetical** so that the system would be able to perform an action (e.g. prompt the user) to find out about this. For example, while in some week the chicken is on special offer and, hence, cheap, the system selects lamb, after it finds out by asking the user that there is a special occasion in this week.

A principled approach to capture the preference requirements, or guidelines, of an application problem, would involve identifying possible combinations of the scenarios expressed directly by the user, which contain conflicting preferences, and prompting or learning from the user further preferences under such combined scenarios and their refinements. This process can 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 expert, or user, these preference requirements.

3. Applications in the Gorgias Argumentation Framework

In this section we briefly overview the Gorgias argumentation framework and indicate how application problems can be formulated as an argumentation theory in it.

Gorgias 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. An **argument**, A, is a set of **argument rules**, denoted by $Premises \rhd Claim$. Such an argument rule links a set of *Premises*, normally given as facts, with the *Claim*: we say that the argument rule **supports** the *Claim*. In an argument, A, through the successive application of its constituent argument rules several claims including a "final" claim are derived and hence supported by A. Argument rules have the syntax of Extended Logic Programming, i.e. rules whose conditions and conclusion (claim) are positive or explicit negative atomic statements without the need to use a second form of Negation as Failure in the conditions[5]. The conclusion of an argument rule can be a positive or negative atomic statement.

In the context of the type of applications described in the previous section, the premises are typically given by a set of conditions describing a scenario and the claim is an option. There are cases, however, where the conditions are themselves defeasible. We call them **beliefs**, and they can be argued for or against, i.e. the claim of an argument can also be a literal on a belief 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. Such object-level arguments can support contradictory claims (e.g. one claim is the other's negation), in which case these arguments **attack** each other. The object-level arguments in the framework are supplemented by **priority arguments** whose purpose is to give a relative strength between arguments. Their role is to tighten the **attack relation** between arguments and sets of arguments, by allowing an argument to attack another only if it is at least as strong as the argument that it attacks. The priority arguments express a local preference between two arguments. They have the same syntactic form of $Premises \rhd Claim$, 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** (i.e. the good quality) of an argument supporting a desirable claim (e.g. an option) occurs between composite arguments. **Composite arguments** are (minimal and closed) set of arguments, $\Delta = (A_1, A_P)$, where, A_1, is a subset of object level argument rules and A_P is a subset of priority argument rules, chosen from the given argumentation theory. Composite arguments thus form the set, $Args$, of (formal) arguments in an abstract argumentation framework, $\langle Args, ATT \rangle$, corresponding to any given Gorgias argumentation theory.

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

[5]There are historical reasons for this as the original motivation of presenting the Gorgias argumentation framework in [27] was to give an argumentation formulation for Logic Programming without Negation as Failure.

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 the papers that introduced *Gorgias* [10, 27].

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 the Gorgias argumentation theory that captures our application problem. The choice of semantics that is predominantly used in the Gorgias framework, is that of **admissible arguments** as the acceptable arguments, namely composite arguments that (a) are not self-attacking and (b) attack back all other arguments that attack them. Admissible arguments thus give, through the options that they support, **"good" solutions** to our problem, i.e. solutions that conform to the guidelines of the problem representation expressed in the argumentation theory. In the special case where there are admissible arguments supporting only one of the possible options of the problem, then this option is regarded as a **"best or optimal" solution**.

3.1. Illustrative Example Continued

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

The knowledge of the various minimal (possibly empty) set of scenario conditions, S^1, in which options are specified as possible ones is represented by object-level arguments, written here in the form[6] $a(i) = S^1 \rhd O_i$, for each option O_i in the subset of preferred options, O^1, of a scenario-based preference statement, $SP^1 = \langle S^1; O^1 \rangle$. Hence, in our example above, with $S^1 = \{main\_shopping\}$, this would give a set of object-level arguments, enabling their corresponding options, as follows:
$a(pork) = \{main\_shopping\} \rhd buy(pork)$
$a(lamb) = \{main\_shopping\} \rhd buy(lamb)$
$a(chicken) = \{main\_shopping\} \rhd buy(chicken)$
$a(fish) = \{main\_shopping\} \rhd buy(fish)$

Given further scenario-based preference statements we can build priority arguments on top of these object-level arguments to capture these statements. For example, SP^2 in our example that expresses the general preference for the cheap foods would be captured, in the specific case of pork and chicken being cheaper than lamb and fish, by the priority argument rules:
$p_{c1}(pork) = \{cheap(pork)\} \rhd (a(pork) > a(lamb))$
$p_{c2}(pork) = \{cheap(pork)\} \rhd (a(pork) > a(fish))$
$p_{c1}(chicken) = \{cheap(chicken)\} \rhd (a(chicken) > a(lamb))$
$p_{c2}(chicken) = \{cheap(chicken)\} \rhd (a(chicken) > a(fish))$
$p_{c3}(pork) = \{cheap(pork)\} \rhd (a(pork) > a(chicken))$
$p_{c3}(chicken) = \{cheap(chicken)\} \rhd (a(chicken) > a(pork))$

Note that the last two priority rules capture the fact that the options of pork and chicken in SP^2 are not comparable or equally preferred. Then, the further refined scenario-based preferences of $SP^3_{m,c,w}$ and $SP^3_{m,c,s}$ for *pork* in the winter and *chicken* in the summer will be represented by the additional *higher-level* priority argument rules:

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

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

These higher level priority rules capture the implicit preference of the more specific scenario-based preferences over the more general ones. Similarly, the other refined scenario-based preference, $SP^3_{m,c,l}$, for a preference to local products amongst the cheaper foods will be represented by higher-level priority argument rules, such as:
$c_l(chicken) = \{local(chicken)\} \rhd (p_{c3}(chicken) > p_{c3}(pork))$

Then, the more refined scenario-based preference, given by $SP^4_{m,c,w,l}$, which considers the combination of the "seasonal" and "local" refinements and prefers local food, is captured with a yet higher-level priority argument rule: $d(chicken) = \{\} \rhd (c_l(chicken) > c_w(pork))$.

Therefore, we see that we can develop a systematic translation of scenario-based preferences, where successive refinements of a scenario give priority argument rules at a **higher level** every time we consider a further refinement in the scenario. Before considering such a general algorithm that transforms hierarchies of scenario-based preferences to Gorgias argumentation theories let us briefly illustrate how the admissibility property of composite arguments is decided in the Gorgias framework.

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 the following two composite arguments, whose argument rules apply (or enabled) under the given application scenario, supporting the option to buy pork or chicken respectively:
$\Delta_1 = \{a(pork), p_{c1}(pork), p_{c2}(pork), p_{c3}(pork)\}$
$\Delta_2 = \{a(chicken), p_{c1}(chicken), p_{c2}(chicken), p_{c3}(chicken)\}$

The priority argument rules that these contain render them stronger than arguments supporting the options to buy lamb or fish. In addition, since there are no priority arguments that are enabled in the current application scenario which can give higher priority to lamb or fish over pork or chicken, the options of lamb and fish are not supported by any relative strong and hence admissible composite argument.

Δ_1 and Δ_2 attack each other 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. We thus have a new argument $\Delta'_1 = \Delta_1 \cup \{c_w(pork)\}$ which is stronger. 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 preference. For this conflict, which comes from $p_{c3}(pork)$ and $p_{c3}(chicken)$, the extra priority rule $c_w(pork)$ in Δ'_1 gives relative priority to the first one and, hence, Δ_2 is attacked by the subset $\{p_{c3}(pork), c_w(pork)\}$ of argument Δ'_1 but Δ_2 cannot attack back, i.e. defend against this attack by Δ'_1. 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 rendering it the only decision/solution that can be admitted in the specific application scenario.

3.2. Transforming Scenario-based Preferences to Argumentation Theories

An application problem description in terms of scenario-based preferences can be automatically transformed into a Gorgias argumentation theory once its SPs have been arranged in a set of SPs hierarchies. Informally, the algorithm which carries this out maps the lowest level of an SP hierarchy, $\langle S^l; O^l \rangle$, into

object level arguments for each of the options in O^1 and then higher-level refinements in the SP hierarchy are mapped into priority arguments over the previous level arguments.

The central part of this translation is Algorithm 1 that maps a *focusing* hierarchy, SP^i $(i = 1, 2, ..., n)$, into arguments. Once we set up, for each option o in O^1 the object level argument: $arg_o^{SP^1} = S^1 \rhd o$, the recursion of this algorithm works to generate the priority arguments at each i^{th} level $(i = 2, ..., n)$ of the hierarchy in the form:

$arg_{o\_over\_o'}^{SP^i} = (S^i - S^{i-1}) \rhd (arg_{o\_over\_o'}^{SP^{i-1}} > arg_{o'\_over\_o}^{SP^{i-1}})$

Note that $(S^i - S^{i-1})$ is the context at the i^{th} level of the hierarchy. The naming, or labeling, of the argument rules used here is $arg_o^{SP^1}$ for the object level rules and $arg_{o\_over\_o'}^{SP^i}$ for the priority rules at each level (with $arg_o^{SP^1} \equiv arg_{o\_over\_o'}^{SP^1}$ for any o').

This elaborate indexing of the argument rules is needed when we extend this algorithm to deal with several scenario-based hierarchies, where it is possible for the same scenario to appear in multiple hierarchies. It is also needed when we generalize this central algorithm to non-focusing scenario-based preference hierarchies. Both these extensions are straight forward to carry out but their details are outside the scope of this paper.

```
1 for (i = 2...n) do
2     for each option o ∈ O^i do
3         for each option o' ∈ O^{i-1} do
4             if complement(o', o) then
5                 for each arg^{SP^{i-1}}_{o_over_o'} do
6                     for each arg^{SP^{i-1}}_{o'_over_o} do
7                         add arg^{SP^i}_{o_over_o'} = (S^i - S^{i-1}) ▷ (arg^{SP^{i-1}}_{o_over_o'} > arg^{SP^{i-1}}_{o'_over_o})
```

Algorithm 1: Central Algorithm for Argument Generation

Algorithm 1 captures the general meta preference of a statement in a specific scenario over statements in more general scenarios. Its application in our running example for the SP focusing hierarchy $\{SP^1_m, SP^2_{m,c}, SP^3_{m,c,w}\}$ will indeed generate the argument rules given above in 3.1.

Through this automatic translation argumentation allows us to solve problems without the need to have an exhaustive representation and look up of scenarios and their preferences. The hierarchies allow the user to express the knowledge in less sentences than the generated rules: the rules in the hierarchy are created invisibly to the user. Argumentation provides a principled way to in effect lookup the scenario-based preferences even in cases where the current information is incomplete and/or contradictory.

3.3. 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 the Gorgias system 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 AI such as non-monotonic learning [27], intra-agent control [28], multi-agent negotiation and dialogue [29, 30], distributed decision making [31], and, reasoning about actions and change [32]. In 2016, a new tool, called Gorgias-B, was released to help the development of applications of argumentation under Gorgias. This tool is presented 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 with the Gorgias argumentation framework and where the Gorgias system was used to implement and evaluate the argumentation-based approach to these problems. We illustrate one of these applications in more detail in section 4.1.

One of the first real-life applications of Gorgias [33] 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 determine the seriousness of the condition of a patient with the possibility of Deep Venous Thrombosis (DVT). Medical expert knowledge was captured as Gorgias argumentation theories of different agents with different expertise. The application was build to provide support to medical practitioners and it was tested on a corpus of 600 patients at a hospital in Cluj-Napoca, Romania.

In the general area of Ambient Intelligence Gorgias has been used to address several decision making application problems. One such application in the more specific area of Ambient Assisted Living [34] was concerned with providing home services for people suffering from cognitive problems, and more particularly from the Alzheimer disease. Personal assistant agents used argumentation to (a) resolve conflicts between competing activities in the user's agenda (e.g. when the user's favorite TV show coincided with the time for taking his/her pills) and to (b) take personalized and context-based decisions in special situations (e.g. to reduce a pill dosage when the weather is particularly hot). This work was part of a larger project, called HERA [35], which was successfully evaluated in real-life trials in two phases. The first phase took place at the Hygeia hospital in Athens, Greece and the second phase at the users' homes.

Gorgias was also applied to the Ambient Intelligence problem of conflict resolution from networked sensors whose information is incompatible with each other [36]. This enables the development of context aware-pervasive services that can adapt to the application environment. The sensors' conflict resolution capability was tested extensively using real-life Web services technology, capable of resolving conflicting data gathered from up to 10 sensors. The authors argued that this shows the potential of argumentation theory to solve real-world problems in services computing.

Recently, we have used Gorgias for addressing another problem of conflict resolution, specifically for resolving diplomatic disputes between countries [37]. In another work [38], Gorgias was used in a context where disputes can arise in a shared environment by many stakeholders, where several legal and other contracts may apply.

Gorgias has been used in applications of network security to provide support tools for authoring, analyzing and managing the firewalls of a corporate network [39]. The management of firewalls becomes a challenging task as they grow through new requirements, imposed by the organization. It requires experienced 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 allowed a direct mapping from the high-level network security policy to the firewall policies. Using *Gorgias* we had an executable firewall configuration, whose compliance to the policy was 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 the existing firewall and to test that it could automatically generate the relative rule orderings that would ensure the correctness, with respect to the given policy, of the firewall configuration. 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 [40]. In another work [41], following a similar approach of formulating firewalls through argumentation, the authors provided a framework where explanations of the behaviour of the firewalls can be documented and exploited by the administrators and users of the network. Recently, Gorgias was used to study another problem in the area of cyber security, namely that of trying to understand cyber attacks on the Internet and attribute to them a likely source [42]. Although this problem is difficult due to the lack of extensive knowledge on the changing nature of cyber attacks, argumentation gives us a principled way of analyzing the problem that helps in its solution.

Another application of Gorgias was Market-Miner, an innovative agent for automated product 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) together with information coming from internal (results of data mining on the corporate database) or external sources (e.g. prices of the competition, weather forecast) and defines appropriate preferences in conflicting scenarios. Briefly, the options considered were at which price to sell a product: normal, high or 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, or a low pricing scenario when the product's type was within a category that the company wanted to hit the competition.

A recent application of *Gorgias* concerned a practical problem of image analysis, namely that of automated discrimination between handwritten and 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. A dynamic decision making process was set up to deal with the possible dilemmas of the agents in conflicting situations, i.e. when both decisions "handwritten" or "printed" are admissible, based on expert default (or generic) and contextual knowledge for solving these conflicts. Each agent is able to propose, in a 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[7], validating its suitability for several real world applications, such as printed text detection, extraction for Optical Character Recognition and electronic document management systems for handling 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 this case, a set of candidate funds is chosen, which are then used by

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

the investor to construct a portfolio according to forecasts and personal preferences. The different approaches applied in the literature, e.g. using linear programming, or applying multi-criteria decision aid methods (see for example [45, 46]), do not provide the opportunity to an investor to define different investment strategies according to personal preferences, or take a decision in an environment with limited information. The investor can take into account the figures and metrics extracted from previous year(s), the market condition, e.g. bull (rising) or bear (falling) 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 and low returns).

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. Thus, the set of options were, $OPTIONS = \{select(Fund), \neg select(Fund)\}$ and the development task was 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 mutual funds. 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)*. With these we can identify several contexts, e.g. type of investor (risk averse and risk tolerant), performance of fund (funds with high Sharpe and Treynor indices), and market context (bull and bear), in which the decision to select or not a fund needs to be considered.

This problem has been captured within a Gorgias argumentation theory constructed along the following lines. First one considers the question of when a fund minimally qualifies to be included or not in a portfolio, i.e. we identify **initial or basic scenarios** that enable the options. 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:
$SP^1_{ns}(Fund) = \langle S^1_{ns} = \{\}; O^1_{ns} = \{\neg select(Fund)\}\rangle$
$SP^1_{s}(Fund) = \langle S^1_{s} = \{highR(Fund)\}; O^1_{s} = \{select(Fund)\}\rangle$,
whose translation into Gorgias argumentation theory leads to the object level arguments:
$a_{ns}(Fund) : \{\} \rhd \neg select(Fund)$
$a_{s}(Fund) : \{highR(Fund)\} \rhd select(Fund)$
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 are then **refined** with the market's characteristics (forecasts) or the investor's attitude. Let's us illustrate some of these and their corresponding scenario-based preferences:

(1) the bear market context, where a mutual fund should not be selected unless it has low risk:
$SP^2_{b,ns}(Fund) = \langle S^2_{b,ns} = S^1_{ns} \cup \{market(bear)\}; O^2_{b,ns} = \{\neg select(Fund)\}\rangle$
$SP^2_{b,s}(Fund) = \langle S^2_{b,s} = S^1_{s} \cup \{market(bear), lowRisk(Fund)\}; O^2_{b,s} = \{select(Fund)\}\rangle$
(2) the risk tolerant investor doesn't select funds unless they have high returns and high risk:
$SP^2_{rt,ns}(Fund) = \langle S^2_{rt,ns} = S^1_{ns} \cup \{investor(risk\_tolerant)\}; O^2_{rt,ns} = \{\neg select(Fund)\}\rangle$

$SP^2_{rt,s}(Fund) = \langle S^2_{rt,s} = S^1_s \cup \{highReturn(Fund), highRisk(Fund), investor(risk\_tolerant)\};$
$O^2_{rt,s} = \{select(Fund)\}\rangle$

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

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

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

The application requires that we also considered **combined** scenarios. For example, the bear market context and risk tolerant investor role can be combined, and, in that case, the risk tolerant investor would accept to select high and medium risk funds (instead of only high). Therefore, we have the additional scenario-based preference:
$SP^3_{b,rt,s}(Fund) = \langle S^3_{b,rt,s} = \{mediumRisk(Fund), investor(risk\_tolerant), market(bear)\};$
$O^3_{b,rt,s} = \{select(Fund)\}\rangle$.

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

The PORTRAIT tool, as it was named, allowed an investor or a fund manager, to select the appropriate funds according to the context (e.g. forecasted situation of the market) and the profile of the investor. Evaluation trails 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]. This had a good effect on the performance of the system 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 methodological approach to acquire, or machine learn, an application's knowledge or requirements, in the form of scenario-based preferences. This new approach has emerged out of the experience of the last decade of applying Gorgias to real-life application problems. It allows the high-level development of applications of argumentation where the interaction with the problem domain expert, or user can occur in terms of the high-level language of the problem that the expert is

familiar with. The experts do not need to have any technical knowledge of computational argumentation in general, or of the Gorgias argumentation framework.

Our approach allows a domain expert (e.g. medical doctor, lawyer, etc) or a user to structure their expertise or personal guidelines using the simple structure of 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. In the case where Machine Learning methods are used, e.g. data analytics, these will operate on the "natural" data of the application (and its operating environment) and will aim to learn the scenario-based preferences of the application, thus filling up a full row of the table. Once we have identified the language of options and description of scenarios the approach follows two basic steps as follows:

STEP 1: The expert (or user) names the columns of the table after the possible options. Then, s/he fills in the first rows, by defining for each option O_i (or set of options $\{O_i, ..., O_n\}$) a *minimal* scenario S^1_α in which this option or a set of options are enabled. For each scenario, defined in a row, 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). We call these scenarios the *basic* (or *initial*) scenarios. They correspond to the first level of scenario-based preferences.

STEP 2: The second step of the approach concerns a process of *conflict analysis* and possible resolution of conflicts. At the end of the previous 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. For the first type of conflict the expert needs to define a next level scenario with new contextual information that narrows down the options range, thus defining a focusing hierarchy. For the second type of conflict the expert declares whether the combined scenario is indeed plausible (i.e it is not inconsistent or completely disjoint) and then considers the preferred options in it.

Some conflicts might still persist and new ones might appear. This process can continue repeating (STEP 2) as long as the expert is able to build further refined scenarios for resolving conflicts. A good practical guideline is to first consider and build hierarchies of scenario-based preferences by iteratively refining scenarios. Once we have such different hierarchies we start considering their combinations, using pairs of hierarchies whose scenarios are different at the lowest levels. We then build hierarchies on top of the combined scenarios. The approach does not impose any restrictions on the expert in making these statements of scenario-based preferences. The only property of the overall preference relation, captured by these statements, 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. Also note that in practice during these two steps the expert or user may introduce new elements in the language of the problem, particularly relations to describe scenarios.

We note that this approach of gathering requirements in the form of scenario-based preferences of increasing specificity is analogue to a form of evidential reasoning where as we increase the evidence we can make "sharper" decisions. The latter has been studied in AI, ranging from an early work in Inheritance Networks [48] to more recent studies of argumentation and evidential reasoning [49, 50].

Table 1 shows an instance of a scenario-based preferences table, being an (intermediate) result at some stage of the process. At the first step the user gives the information in $S^1_{1,3}$ in row 1 of the table and $S^1_{1,2,8}$

Table 1
Example of scenario-based preferences on conflicting options

row #	Scenarios \ Options	O_1	O_2	O_3	O_4	O_5	O_6
1	$S^1_{1,3} = \{x_1, x_3\}$	X			X		X
2	$S^2_{1,3,6} = \{x_1, x_3, x_6\}$	X					X
3	$S^3_{1,3,6,y} = \{x_1, x_3, x_6\} \cup \{y\}$	X					
4	$S^3_{1,3,6,ny} = \{x_1, x_3, x_6\} \cup \{\neg y\}$						X
5	$S^1_{1,2,8} = \{x_1, x_2, x_8\}$		X	X		X	
6	$S^2_{1,2,8,9} = \{x_1, x_2, x_8, x_9\}$		X				
7	$S^2_{1,2,3,8} = \{x_1, x_2, x_3, x_8\}$		X	X		X	
8	$S^3_{1,2,3,8,z} = \{x_1, x_2, x_3, x_8\} \cup \{z\}$				X		X

in row 5 (for the subscript of S we will use the indexes i for each x_i in S). These are two basic scenarios and form the lowest level of two hierarchies. At the next step of conflict analysis and resolution each of the basic scenarios is refined. $S^1_{1,3}$ is refined with x_6 that narrows down the selected options to O_1 and O_6 (row 2). $S^1_{1,2,8}$ is refined with x_9 that narrows down the selected options to O_2 (row 6). Moreover, the table shows the case of a combined scenario where the conditions of both of the basic scenarios hold at the same time. This is at row 7 which requires that the options of $S^1_{1,2,8}$ are preferred in this combined scenario.

Given these rows, i.e. rows 1,2,5,6 and 7, we see that there are still conflicts as, for example, there is a conflict between options O_1 and O_6 under the scenario $S^1_{1,3,6}$ in row 2. The expert can then examine whether s/he can discriminate between the conflicting options by considering refined scenarios, iterating again STEP 2 of the methodology. When this is possible new rows with refined scenarios are added in the table. For example, a new row (row 3) is added in table 1 resulting in a focusing hierarchy based on scenario $S^1_{1,3}$ and whose corresponding scenarios are those of rows 1 to 3. Another focusing hierarchy based on $S^1_{1,3}$ is in rows 1, 2 and 4: $S^1_{1,3}$, $S^2_{1,3,6}$, $S^3_{1,3,6,ny}$. A focusing hierarchy based on $S^1_{1,2,8}$ is in rows 5 and 6 of the table: $S^1_{1,2,8}$, $S^1_{1,2,8,9}$. In row 6 the option O_2 is preferred over O_3 and O_5, when the extra scenario information x_9 is added to the scenario $S^1_{1,2,8}$.

The table also shows in row 7 that the options O_2, O_3, O_5 are preferred over options O_1, O_4, O_6 in the consistent combined scenario $\{x_1, x_2, x_3, x_8\}$ of the basic scenarios in rows 1 and 5. 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 the additional information "z" holds, thus refining the combined scenario $\{x_1, x_2, x_3, x_8\}$. This is seen in the last row of the table. This combined scenario defines two more hierarchies: a) $S^1_{1,3}, S^2_{1,2,3,8}, S^3_{1,2,3,8,z}$ and b) $S^1_{1,2,8}, S^2_{1,2,3,8}, S^3_{1,2,3,8,z}$. Note that we do not have a combined scenario of rows 3 and 4 as it would contain both y and $\neg y$ and such a combined scenario is not plausible.

It is important to note that the tables of our approach 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 expand them, we also introduce a conflict analysis step to ascertain that 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 they first appear. At a later stage, when extra knowledge has been acquired, the user can discriminate between existing conflicts.

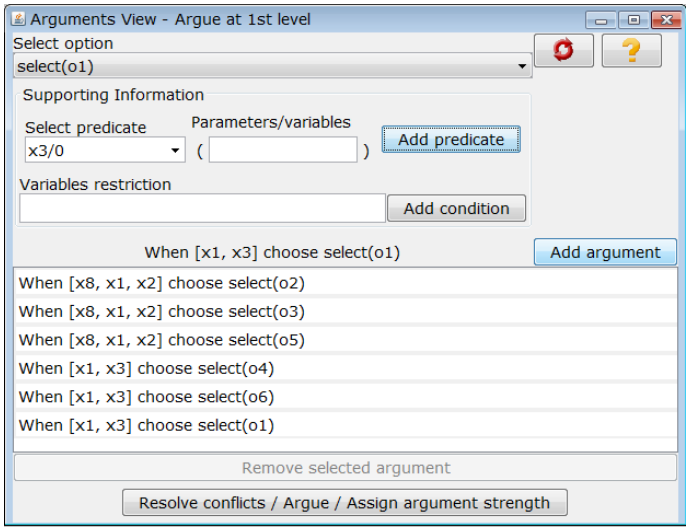

Fig. 1. *Gorgias-B*: The arguments view (step 1 of the methodology)

5.1. Gorgias-B: Authoring Tool for Scenario-based Preferences

Gorgias-B is a Java-based tool built on top of *Gorgias* for supporting the development of *Gorgias* argumentation theories[8]. *Gorgias-B* has two important roles in the development of applications of argumentation. 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 us to execute scenarios, i.e. to find out which options are admissible in a given scenario and hence the user can test the generated 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 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 declared options, corresponding to the first step of the process of capturing initial scenarios. For example, in the figure we see that the user has selected option, O_1, i.e. the option predicate *select(o1)*, and has added the predicate conditions *x1* and *x3*, to form an object-level argument corresponding to the first "X" of row 1 of Table 1. On the left of the "Add argument" button the user can see the scenario based preference that is ready to be added, e.g. "When $[x1, x3]$ choose $select(o6)$". 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 preferred set of options of the row. The six object level arguments are visible in the list at the bottom half of Figure 1.

Continuing with STEP 2 of the process, the user clicks the button "Resolve conflicts/Argue/Assign argument strength" which opens a new dialogue (see Figure 2). *Gorgias-B* identifies scenarios with conflicting options and the user can work on these by selecting (using the controls on the right of the

[8] *Gorgias-B* (http://gorgiasb.tuc.gr) runs under 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.

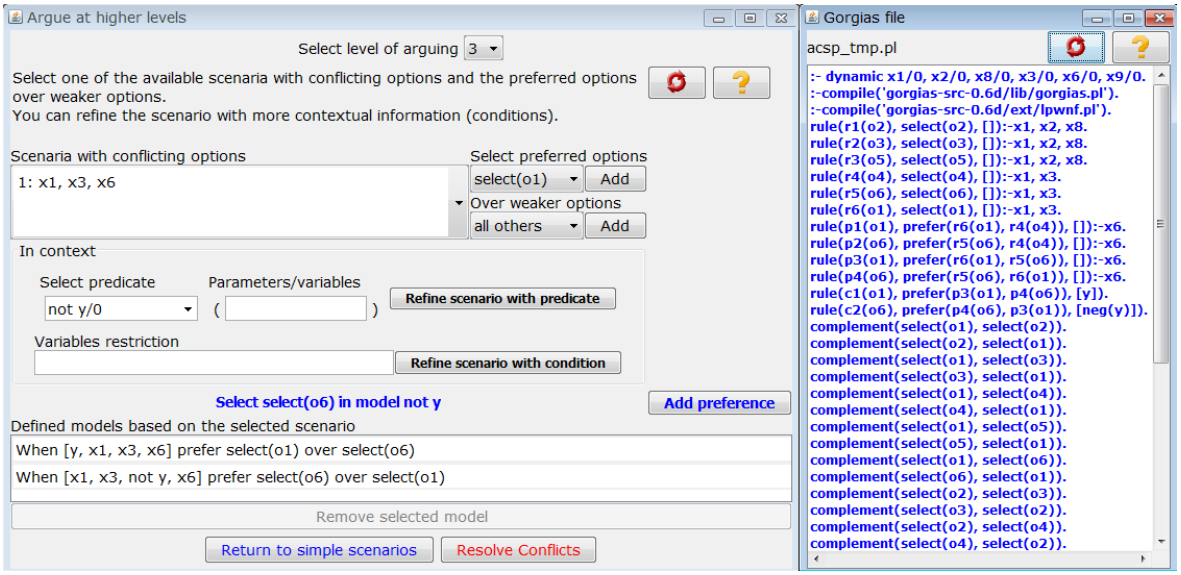

Fig. 2. *Gorgias-B*: Arguing at higher levels (step 2 of the methodology, second iteration)

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, thus constructing refinements of the scenario. These preference statements appear in the main window as binary preferences or priorities between pairs of options.

In this way, the tool allows the user to enter scenario-based preferences at higher levels of scenario refinements. For example, Figure 2 shows the conflicting scenario $\{x_1, x_3, x_6\}$, where the preference corresponding to row 3 for the refined scenario, $S^3_{1,3,6,y}$ (see Table 1), is inserted, generating the first priority argument of the two that appear in the box at the bottom of Figure 2. Then the user inserts the preference corresponding to row 4 for the refined scenario $S^3_{1,3,6,ny}$ (seen in Figure 2 as the second priority statement in the bottom box). Note that the user can navigate the levels of the different hierarchies either by selecting the level at the top of the window or by clicking the buttons at the bottom which are only enabled if the hierarchy has previous or next levels. If a conflict exists at the current level the "Resolve conflicts" is enabled (and shown in red).

By pressing this "Resolve Conflicts" button at the bottom we can move to a next level of conflict analysis corresponding to a new iteration of STEP 2 of our methodological approach. In this way, the user adds the scenario-based preferences identified at the higher-level steps. Figure 3 shows this for row 7 of Table 1, with the scenario $S^2_{1,2,3,8} = \{x_1, x_2, x_3, x_8\}$. 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.

Row 8 of $S^3_{1,2,3,8,z}$ 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 entered in *Gorgias-B*, 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 $\{x_1, x_2, x_3, x_8, z\}$

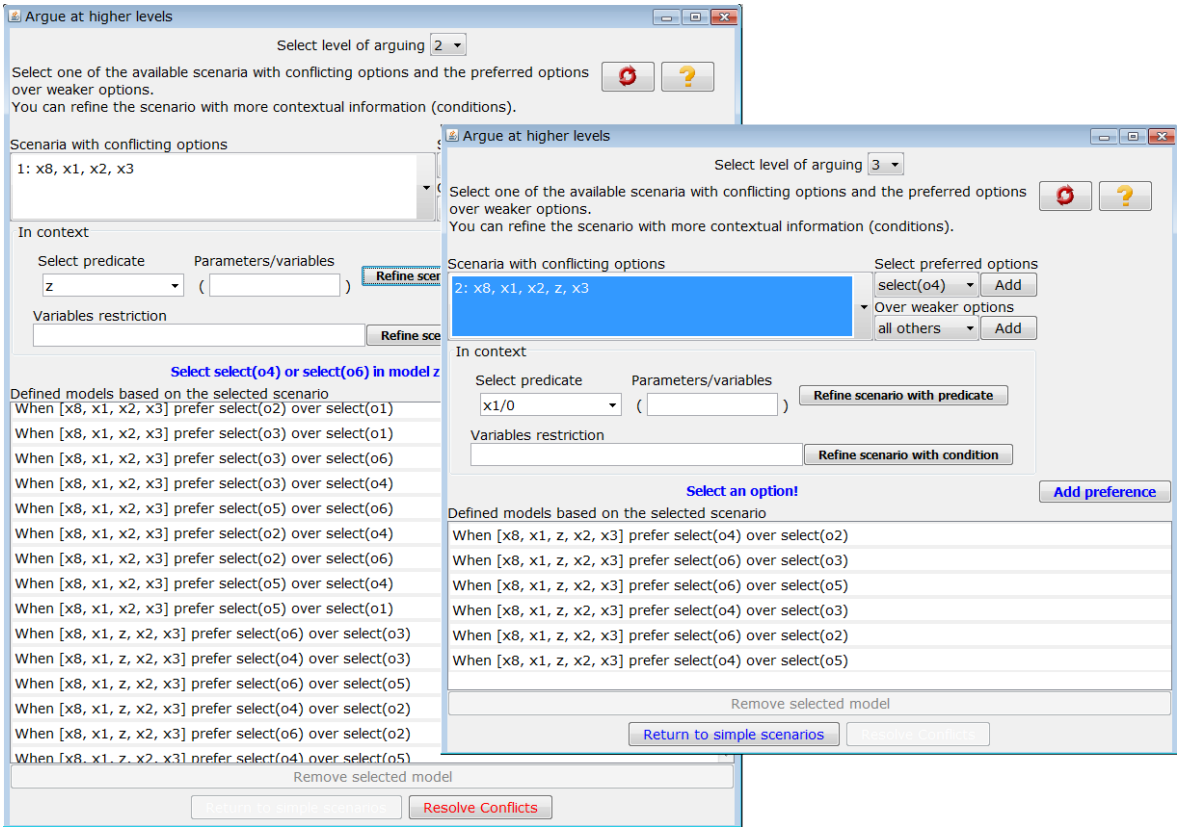

Fig. 3. *Gorgias-B*: Arguing at higher levels (combination in second level and refinement in third)

(see right hand side of Figure 3). *Gorgias-B* sets automatically these higher-level preferences as the most specific context is automatically given preference. Of course, the user can delete unwanted preferences and/or add her own.

The “Resolve Conflicts” button of the level "2" screen (left screen) in Figure 3 is enabled. This means that in the current scenario *Gorgias-B* has still conflicting preferences that could be resolved at the next level. As soon as the user clicks this button, s/he 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.

Finally, the “Gorgias file” view on the right hand side of the dialogues, e.g. as seen in Figure 2, is optional and allows the user to see the *Gorgias* argumentation theory code that is automatically generated as the scenario-based preferences are entered.

6. Current Development of Applications

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

6.1. Eye-clinic (first level) Support

This application concerns the development of a system that can provide a first level support in an eye-clinic by analyzing the symptoms presented by the patient, possibly requesting extra information, 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 help the receptionist/nurse to make a preliminary (i.e. before a doctor's examination) diagnosis in order to plan the appointments of the doctors based on the severity of the patient's condition. The nurse would supply the system with observable (or declared by the patient) facts (e.g. symptoms such as red eye, painful eye, etc.) thus building a current scenario of interest. The system would also prompt the nurse to request additional information when needed (e.g. to measure the intra-ocular pressure or ask the patient about past history) in order to make a more informed diagnosis.

The development of our Eye-clinic support system was based on a collaboration with a highly qualified ophthalmologist doctor of a public hospital in Paris (France) (see [51]). The system is based on the set of known ocular diseases (there are more than 80), which constitute the options in the application's decision problem. The task is to select the most appropriate ones, given the information gathered from the patient. Options are represented by predicates of the form: $ds(Name\_of\_disease, Severity)$. For illustrating this application we will consider a sub-case where we are concentrating only on four diseases as the set of options. This is given by the following, where the severity of the disease is represented by a number whose higher value indicates higher severity:

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

As mentioned above, the scenario information consists of observable facts in various categories, such as the area of the symptom, called *zone* (e.g. the eye), or the category of *symptoms* (e.g. red eye, painful eye) or the *context* (e.g. ocular surgery). We also have other hypothetical information such as whether the patient had (or not) a disease previously. 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. Basic, or initial, scenarios with minimal information allow for several diseases to be initially chosen. 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 scenarios "z" stands for "zone", "s" for "symptom", "c" for "context", "oc_sur" for "ocular surgery", and "vis_dist" for "visual disturbance". The expert has identified one initial scenario, $S^1_{ze,sr}$, leading to all possible diseases. 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. The refinement from $S^1_{ze,sr}$ to $S^2_{ze,sr,sv,sp}$ narrows down the options but the subsequent refinement of the scenario to $S^3_{ze,sr,co,sv,sp}$ reinstates D_4 as the preferred option. The last two rows are two different refinements of $S^2_{ze,sr,sv,sp}$.

Our methodology has allowed us to structure 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 knowledge that has been acquired. 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

Table 2
Example of eye clinic application

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

on the combination of basic scenarios when she had to compare and discriminate between conflicting diseases in various existing scenario-based preferences. A prototype is under development and we have already implemented more than 3,000 rules representing object and priority arguments. A commercial product will be subsequently developed to be deployed in the collaborating eye clinic.

6.2. Data Access and Data 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 a data sharing agreement (DSA). Recent works [38, 52] have applied an argumentation approach within the Gorgias framework for data access and sharing in the health sector. The *MEDICA* [9] [53] system was developed for the purpose of determining the level of access to patients' data records, according to an EU country's law on medical data access. There are six different access levels, ranging from *full access* to various types of *limited access* and to *restricted* or *no access*. All decisions on the level of access reached by the *MEDICA* system are explained to the user by reference to the relevant articles of legislation, which are, in fact, the basis for the object and priority level arguments that support the reached decision. The system also supports requests from users wanting to gain higher level of access by indicating the extra information that needs to hold, when indeed such higher level access is possible.

More specifically, in this problem the set of Options is captured by the predicate $access(P, F, Type)$, where P names the requester, F names a data file and $Type$ takes values in: {*Full, LimitedPlus, Limited-PlusReadOnly, Limited, Suspended, No*}, corresponding to the levels of access stipulated by the law.

Table 3 shows a representative case of scenario-based preferences extracted from the legislation. Normally none has access to a medical file. The owner of the file has full or suspended access for personal use. If the file is suspended then the owner has only suspended access. A doctor can have limited plus read only access for treatment purposes. However, even in this case, if the owner of the data is dead access is not granted.

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 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. Subsequently, we will proceed to evaluate the system through a 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 across policies where we need to understand how one policy's preference takes precedence over the others. Such is the situation

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

Table 3

A scenarios table. The decision predicate is $access(P,F,T)$. Depending on the value of variable T we have the access levels: $AP_1 : full, AP_2 : limited\_plus, AP_3 : limited\_plus\_read\_only, AP_4 : limited, AP_5 : suspended, AP_6 : no$. Scenario subscripts for contexts: *file*: f, *owner*: o, *personal use*: pp, *suspended*: s, *doctor*: d, *treatment*: pt, *dead owner*: od

Scenarios \ Access Policies	AP_1	AP_2	AP_3	AP_4	AP_5	AP_6
$S^1_f = \{file(F)\}$						X
$S^2_{f,o,pp} = \{file(F)\} \cup \{owner(P), purpose(personal)\}$	X				X	
$S^3_{f,o,pp,s} = \{file(F), owner(P), purpose(personal)\} \cup \{suspended(F)\}$					X	
$S^2_{f,d,pt} = \{file(F)\} \cup \{doctor(P), purpose(treat)\}$			X			
$S^3_{f,d,pt,od} = \{file(F), doctor(P), purpose(treat)\} \cup \{owner(F,O), dead(O)\}$						X

in the data sharing landscape (see e.g. [38, 54]). Data is generated in independent and distributed nodes and access to it by different stakeholders is governed by a diverse set of policies. As stakeholders can be independent entities, with their own policies, there are typically several cases where their policies will conflict and in that case an arbitrator is needed to handle the conflicts.

Recently, we proposed a method [55] based on Gorgias for managing such conflicts across different stakeholders' policies. The main idea is to build an arbitrator *meta policy* over the individual policies. 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. the Fire Service) such a meta policy can be built from data sharing statements of the form:

> "The personal and legislation policies are preferred over others. Among them the personal policy is preferred. However, if the person is a victim in an accident scene, the fire service policy is preferred over the personal one. If the owner of the data is hospitalized, the hospital policy is preferred."

This approach allows the arbitrator to be agnostic with regard to the individual reasons why a policy allows or denies access and at the same time be capable to arbitrate on such a matter. We are also working on applying such meta policy arbitration to regulate energy consumption in smart green buildings [55].

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, during the last decade, of applying the argumentation framework of *Gorgias* to solve real-life problems. The proposed approach allows the modular development of real-life applications, where requirements are processed 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. Our work shows in practical terms how argumentation is suited to solve problems under incomplete and incompatible information. Argumentation provides a principled and theoretically well founded way to solve such problems providing at the same time explanations for the solutions. The approach is sufficiently general to allow the development of applications in any, of the many existing argumentation frameworks (e.g. [8, 56, 57]), that support conditional and hierarchical forms of preference.

The approach can be improved in several ways. An important development of the authoring 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 application development approach various methods of Machine Learning. For example, scenario-based preferences can be extracted by data analytics on past cases of the application without the need for their explicit stipulation by the domain expert/user. Similarly, by including a feedback process during the operation of the system we can learn new or refined preferences from experience. This feedback can be linked to the provision of explanations that argumentation systems can offer for their operation and decisions. Thus, we are working to extend the explanation facility of our tools to so as to be able to give these in a natural form familiar to the users and to be able to engage in dialogues with the users. In fact, we believe that our approach shows how argumentation facilitates the integration of sub-symbolic Machine Learning methods with Symbolic Computation to build systems that can have a cognitively natural interface with human developers and users.

We are also experimenting with the development of personal 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* [58] 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, Smarter Electricity and Argumentation Theory, in: *Combinations of Intelligent Methods and Applications: Proc. of the 4th International Workshop, CIMA 2014, Limassol, Cyprus, November 2014*, I. Hatzilygeroudis, V. Palade and J. Prentzas, eds, Springer Int. Publishing, Cham, 2016, pp. 79–95.

[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 11th Int. Conf. on Logic Programming, Santa Marherita Ligure, Italy, June 13-18*, 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] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] I.A. Letia and M. Acalovschi, Achieving Competence by Argumentation on Rules for Roles, in: *Engineering Societies in the Agents World V: 5th Int. 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, 2005, pp. 45–59.

[34] J. Marcais, N. Spanoudakis and P. Moraitis, Using Argumentation for Ambient Assisted Living, in: *Artificial Intelligence Applications and Innovations: 12th INNS EANN-SIG Int. Conf., EANN 2011 and 7th IFIP WG 12.5 Int. Conf., 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. doi:10.1007/978-3-642-23960-1_48.

[35] 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.
[36] 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.
[37] 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.
[38] 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.
[39] 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.
[40] 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.
[41] 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.
[42] E. Karafili, L. Wang, A. Kakas and E. Lupu, Helping Forensics Analysts to Attribute Cyber-Attacks: An Argumentation-Based Reasoner, in: *Proceedings of the 21st Int. Conf. on Principles and Practice of Multi-Agent Systems (PRIMA 2018)*, 2018.
[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 Int. Conf., 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.
[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] J.F. Horty, R.H. Thomason and D.S. Touretzky, A skeptical theory of inheritance in nonmonotonic semantic networks, *Artificial Intelligence* **42**(2) (1990), 311–348, ISSN 0004-3702. doi:https://doi.org/10.1016/0004-3702(90)90057-7.
[49] H. Prakken, A study of accrual of arguments, with applications to evidential reasoning, in: *The Tenth International Conference on Artificial Intelligence and Law, Proceedings of the Conference, June 6-11, 2005, Bologna, Italy*, 2005, pp. 85–94.
[50] B. Verheij, Proof with and without probabilities - Correct evidential reasoning with presumptive arguments, coherent hypotheses and degrees of uncertainty, *Artif. Intell. Law* **25**(1) (2017), 127–154.
[51] 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.
[52] 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.
[53] 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.
[54] F. Martinelli, I. Matteucci, M. Petrocchi and L. Wiegand, A Formal Support for Collaborative Data Sharing, in: *Multidisciplinary Research and Practice for Information Systems, CD-ARES 2012, Prague, Czech Republic, August 20-24*, G. Quirchmayr, J. Basl, I. You, L. Xu and E. Weippl, eds, Springer, 2012, pp. 547–561. ISBN 978-3-642-32498-7.
[55] N. Bassiliades, N.I. Spanoudakis and A.C. Kakas, Towards Multipolicy Argumentation, in: *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, SETN '18, ACM, New York, NY, USA, 2018.
[56] 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.
[57] 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.
[58] 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.