

Argumentation for Normative Reasoning

Nir Oren and Michael Luck¹ and Timothy J. Norman²

Abstract. An agent’s behaviour is governed by multiple factors, including its beliefs/desires/intentions, its reasoning processes and societal influences acting upon it, such as norms. In this paper we propose an extensible argumentation inspired reasoning procedure, and show how it may be used to perform normative reasoning. The language used by our procedure is built around defeasible, non-monotonic rules called argument schemes. The evaluation of the interactions between argument schemes and predicates is performed using a novel argumentation based technique. We show how issues such as normative conflict, priorities over norms, and the effects of norms may be represented using the framework, and how the agent may use these to reason effectively.

1 Introduction

An increasingly popular way of declaratively controlling agent behaviour is through the use of norms. Most commonly, a set of obligations and prohibitions are imposed upon an agent, constraining its behaviour accordingly. Different approaches define these norms in different ways. In the simplest case, an obligation can be seen as a hard constraint, with the system entering an undefined state if the norm is violated. More flexible systems treat norms as soft constraints, but as flexibility increases, difficult questions arise in areas including normative reasoning, verification, and semantics, as well as how norms interact with each other.

In particular, additional complications arise when normative conflicts occur, with an agent having to select which norms to honour, and which to ignore. Ultimately, the agent’s actions are governed by its preferences, knowledge (including beliefs, desires and intentions), the norms affecting it, and the state of the environment. Several efforts to address this issue have been proposed, which take these various factors into account in different ways.

One strategy espoused by some (including the philosopher John Pollock [8]) follows the approach that humans appear to use when faced with multiple choices; namely to engage in an *internal dialogue* and act based on its outcome. For example, if I am to decide whether to play a game or write a paper, I would weigh up the pros and cons of each of these actions, in the context of my obligations, e.g. when the paper is due, and how much I like my job. This weighting process may create additional reasons to pick one action over another, and may cause other reasons to no longer be applicable.

Pollock’s work was intended to apply to any form of practical reasoning, and did not pay particular attention to dealing with norms. His internal dialogue based representation of practical reasoning, instantiated via an argumentation procedure, is highly appropriate for reasoning about norms because norms typically constrain desired behaviour, leading to the need to resolve (internal) conflicts so as to

allow an agent to determine whether to comply with its norms. Our reasoning framework, while simpler than Pollock’s, contains the features necessary to reason about normative issues. It is based on the emerging AIF standard [3], and provides support for normative concepts via the idea of argument schemes. Argument schemes represent defeasible, possibly non-deductive, rules of inference, and are intended to capture common patterns of argument. They may be general, or domain-specific.

In this paper, therefore, we use argument schemes to represent reasoning rules. We present a number of argument schemes that can be used to reason about normative concepts. By representing its knowledge using these argument schemes, and using results from argumentation theory, an agent is able to infer, from the interactions between argument schemes, how to act on the basis of its norms, and whether any of its norms should be ignored. Our approach is able to naturally deal with normative conflict and, due to its non-monotonic nature, is easily able to handle cases where an agent is presented with additional information. Apart from the formal introduction of normative argument schemes, our main contribution thus revolves around the framework’s ability to aid an agent in resolving normative conflict. Thus, we begin the next section by introducing argument schemes, after which we show how an agent may reason about their interactions. After providing an example showing the framework in action, we conclude the paper by examining related research and proposing further extensions to the work presented here.

2 Argument Schemes and Information

As mentioned in the introduction, agents using our framework reason about the world using inference rules called argument schemes. We begin this section by informally describing these argument schemes. Later, in Section 4, we show how these argument schemes may be used for normative reasoning.

An argument scheme represents a (possibly non-deductive) rule of argument. Argument schemes consist of three components: a set of premises, a set of conclusions, and a set of other argument schemes which may be *undercut* by this scheme (an undercut represents a reason for not being allowed to use the scheme, and is described in more detail later). Premises and conclusions refer to concrete elements of the environment, form the basis of our language, and are represented using Prolog-like predicates. As per Prolog convention, we assume that the first letter of a variable is capitalised, while a constant’s first letter is lowercase. Unbound predicates are predicates containing variables; when we refer to predicates, we mean predicates containing no unbound variables. We call the set of all unbound predicates *UPS*, while the set of predicates is labelled *PS*.

We define a function $subst(A, B)$, which returns predicate A unified according to the mapping B , where $A \in UPS$, and B is a mapping between variables and unbound predicates. Predicates are used

¹ King’s College London, UK, email: nir.oren,michael.luck@kcl.ac.uk

² University of Aberdeen, Scotland, email:t.j.norman@abdn.ac.uk

as the basis for knowledge representation within an agent’s knowledge base, and are used to represent both contested and uncontested facts. These facts are used as inputs to argument schemes, and may also form as a result of the application of such a scheme. AIF refers to such predicates as *information*, and we will adopt this terminology.

As stated previously, an argument scheme represents a rule of inference (and usually refers to a repeatedly used form of argument). Argument schemes operate on specific types of information, and thus make use of unbound predicates; they are applicable when specific conditions hold, namely when all their premises are present. The application of an argument scheme results in some conclusions being drawn. Argument schemes may also influence the application of other schemes. Pollock gives the following example of an argument scheme undercutting another argument scheme (i.e. preventing the other scheme coming into force). Here, the application of the second argument scheme nullifies (undercuts) the first scheme’s application.

1. If an object looks a certain colour, it is that colour.
2. If a light of a certain colour shines on an object, that object will take on the light’s colour (even if the object is not that colour).

Strong parallels exist between argument schemes and default reasoning, particularly when dealing with issues such as burden of proof and critical questions [10]. Premises to an argument scheme may thus be required, or be default (i.e. only the explicit presence of the negation of a default may cause a scheme to not be applied). Since our language does not explicitly cater for negation, two types of conclusions may exist for a given argument scheme, namely those conclusions supported by the scheme, and those attacked by the scheme. Formally then, we may define an argument scheme as follows:

Definition 1 (Argument Scheme) *An argument scheme is a structure of the form*

$$AS = \langle ASDefaults, ASPremises, ASSupports, ASAttacks, ASUndercuts \rangle$$

where $ASDefaults, ASPremises, ASSupports, ASAttacks \in 2^{UPS}$. Furthermore, a predicate may only appear in one of $ASDefaults, ASPremises, ASSupports$ and $ASAttacks$. If a variable v appears in an unbound predicate within $ASSupports$ or $ASAttacks$, it must also appear within $ASDefaults$ or $ASPremises$. $ASUndercuts$ is a set of undercutting bindings.

$ASDefaults$ represents the set of defaults which must not hold for an argument scheme to be applied, while $ASPremises$ represents those pieces of information that must hold. $ASSupports$ consists of those conclusions supported by the argument scheme, and $ASAttacks$ stores the conclusions that are attacked by the scheme.

Definition 2 (Undercutting bindings) *An undercutting binding is a pair $\langle AS, UBMMapping \rangle$ where AS is an argument scheme, and $UBMapping$ is a mapping from the variables found in $(ASPremises \cup ASSupports \cup ASAttacks)$ to UPS .*

As an example, consider the argument scheme “if A implies C , and B implies C , then, by accrual, C is to be strongly believed”. Formally, this argument scheme could be represented as follows:

$$\{\}, \{A, B, \text{implies}(A, C), \text{implies}(B, C)\}, \\ \{C, \text{strongly}(C)\}, \{\}, \{\}$$

Such an argument scheme implicitly assumes that the implication may accrue. Another argument scheme, together with its associated

undercutting bindings may be defined to prevent this scheme from being applied in situations where accrual may not occur³:

$$\{\}, \{\text{notAccrue}(X, Y)\}, \{\}, \{\}, \\ \{\{\{A, B, \text{implies}(A, C), \text{implies}(B, C)\}, \\ \{C, \text{strongly}(C)\}\}, \{\}, \{\}, \{\{A, X\}, \{B, Y\}, \{C, C\}\}\}$$

Note that there is no specialised representation for negation in our framework. The following schemes are thus present in most systems:

$$\langle \{\}, \{A\}, \{\}, \{\text{not}(A)\}, \{\} \rangle \\ \langle \{\}, \{\text{not}(A)\}, \{\}, \{A\}, \{\} \rangle$$

An argument scheme may be used, together with some information, to generate new information. The application of an argument scheme in this way results in an instantiated argument scheme, and is achieved via a process of unification.

Definition 3 (Instantiated Argument Scheme) *An instantiated argument scheme is a tuple $IAS = \langle AS, Mapping \rangle$ where AS is an argument scheme, and $Mapping$ is a mapping such that,*

$$\forall U \in ASPremises, \text{subst}(U, Mapping) \in KB \\ \forall U \in ASDefaults, \text{subst}(U, Mapping) \notin KB \\ \forall U \in ASSupports \cup ASAttacks, \text{subst}(U, Mapping) \in PS$$

3 Evaluating Arguments

The process of argument not only generates arguments, but also links information in a specific way, with some arguments supporting others, and other arguments and pieces of information attacking each other in various ways. Given a set of arguments, an agent must be able to determine which of these are, in some sense, admissible. That is, which arguments may be deemed to “hold” given the interactions within the set. The most influential work in this area is probably that of Dung [4]. Here, arguments are treated as abstract entities, with no attention paid to their content. The only attribute associated with arguments is that they require the ability to attack other arguments. Given a set of arguments, and a binary attack relation, Dung neatly classifies the sets of arguments that a rational reasoner deems to be admissible at the end of an argument.

Various extensions to Dung’s work have been proposed (e.g. [2, 1]), and in this paper, we make use of Oren’s work on abstract evidential reasoning [6], which we now describe. An evidential argument system stores the arguments and records the way they may interact with each other. Two types of interactions may exist: attacks, and supports. Oren’s work investigated which sets of arguments may be considered “admissible”; that is, which sets make sense to some sort of rational reasoner, given the interactions between arguments. In his work, an argument is only considered admissible if there is an unbroken chain of support from some sort of universally accepted “evidence argument” to the argument. A chain may be broken if it is successfully attacked by some other supported argument.

Definition 4 (Evidential Argument System) *An evidential argument system is a tuple (A, R_a, R_e) where A is a set of arguments, R_a a relation of the form $(2^A \setminus \{\eta\}) \times A$ and R_e a relation of type $2^A \times A$. Additionally, $\nexists z \in 2^A, y \in A$ such that $zR_a y$ and $zR_e y$.*

³ It is also possible to represent this by introducing “virtual” information and using the $ASDefault$ property.

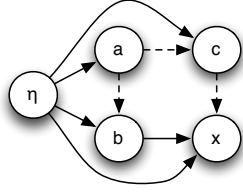


Figure 1. An evidential argument system. Solid arrows represent support, while dashed arrows represent attacks between arguments.

The R_e and R_a relations respectively encode evidential support and attacks from a set of arguments, to an argument⁴. η represents a special argument, representing unquestionable support from the environment, and can also be used to represent defaults. As an example, consider the following arguments:

- a It was dark. When it is dark, a witness's statement could be wrong.
- b Witness b made the statement that the bird flew. A witness' statement can be viewed as evidence.
- c Witness c made the statement that the bird did not fly. A witness' statement can be viewed as evidence.
- x We know that birds can normally fly, and thus given some evidence, we may claim that the bird flew.

This would result in the following evidential argument system, as illustrated graphically in Figure 1.

$$\begin{aligned} &< \{a, b, c, x\}, \\ &\{(\{a\}, c), (\{a\}, b), (\{c\}, x)\}, \\ &\{(\{\eta\}, a), (\{\eta\}, b), (\{\eta\}, c), (\{\eta\}, x), (\{b\}, x)\} > \end{aligned}$$

Within an evidential argument framework, it is necessary for an argument to be supported by some other argument for it to ultimately be admissible. This immediately raises bootstrapping concerns as some initial argument has to be supported for any other argument to eventually be supported. We overcome this problem with the special η argument:

Definition 5 (Evidential Support) An argument a is supported by a set S iff

1. $SR_e a$ where $S = \{\eta\}$ or
2. $\exists S' \subset S$ such that $S'R_e a$ and $\forall x \in S', x$ is supported by $S \setminus \{x\}$

S is a minimum support for a if there is no $S' \subset S$ such that a is supported by S' .

Assume, in the example above, that no direct link exists between η and x . Then x is supported by the set $\{\eta, b\}$. Given the notion of evidential support, we may define an *evidence-supported* attack:

Definition 6 (Evidence-Supported Attack) A set S carries out an evidence-supported attack on an argument a if

- $XR_a a$ where $X \subseteq S$, and,
- All elements $x \in X$ are supported by S .

An evidence-supported attack by a set S is minimal iff there is no $S' \subset S$ such that S' carries out an evidence-supported attack on a .

⁴ Attacks and supports from sets of arguments allow us to represent conjunctive arguments [5].

We will usually write *s-attack* when referring to an evidence-supported attack. A *s-attack* represents an attack backed up by evidence or facts. Within the example, Arguments a and c are both supported by η ; a thus *s-attacks* b , and c *s-attacks* x .

Support for an argument is clearly one requirement for acceptability, but it is not enough. Following Dung, an argument should be acceptable (with respect to some set) if it is defended from attack by that set. The question arises as to whether all attacks should be defended against, or only *s-attacks*. Since the framework focuses on *s-attacks*, we choose the latter option, allowing an argument to be defended from attack by either having the attack itself attacked, or by having any means of support for the argument attacked by the defending set.

Definition 7 (Acceptability) An argument a is acceptable with respect to a set S iff

1. S is a support for a .
2. Given a minimal *s-attack* $X \subseteq 2^A$ against a , $\exists Y \subseteq S$ such that $YR_a x$ where $x \in X$ so that $X \setminus \{x\}$ is no longer a *s-attack* on a .

An argument is thus acceptable with respect to a set of arguments S if any argument that *s-attacks* it is itself attacked (either directly, or by being rendered unsupported) by a member of S . The set S must also support the acceptable argument. In the example, the x is acceptable with respect to the set $\{\eta, a\}$. However, acceptability here is lacking in one respect in that it does not examine whether elements of S might themselves be attacked by other arguments (which would prevent S from supporting a). The concept of admissibility overcomes this issue, but to define it, we need two further notions.

Definition 8 (Conflict free and Self Supporting Sets) A set of arguments S is conflict free iff $\forall y \in S, \nexists X \subseteq S$ such that $XR_a y$.

A set of arguments S is self supporting iff $\forall x \in S, S$ supports x .

Admissibility may now be defined as follows, so that in the example, $\{\eta, a\}$ is an admissible set.

Definition 9 (Admissible Set of Arguments) A set of arguments S is said to be admissible iff

1. All elements of S are acceptable with respect to S .
2. The set S is conflict free.

We are now in a position to define what sets of arguments a rational reasoner may find consistent. Dung named these sets “extensions”. Multiple types of extension exist, representing, among others, sets of arguments that credulous and sceptical reasoners may find consistent. For example, given the arguments x = “The man is guilty because of reasons a, b, c ”, and y = “the man is innocent because of reasons d, e, f ”, where $a \dots f$ are independent unrelated arguments, a sceptical reasoner would deem x, y inadmissible, as it has no way of choosing between them. A credulous reasoner would instead say that there are two scenarios, one where x is true, and one where y is true, and would not be able to choose between them.

Definition 10 (Extensions) An admissible set S is an evidential preferred extension if it is maximal with respect to set inclusion. That is, there is no admissible set S' such that $S \subset S'$.

An evidential grounded extension of an evidential argument framework EA containing the set of arguments A is the least fixed point of F_{EA} . Where

$$F_{EA} : 2^A \rightarrow 2^A$$

$$F_{EA}(S) = \{a | a \text{ is acceptable with respect to } S\}$$

```

Given a set of predicates  $P$ 
Given a set of instantiated argument schemes  $I$ 

1  For any  $i = \langle \langle iASDefs, iASPrems, iASSupps,
   iASAtts, iASUCuts \rangle, iMap \rangle \in I$ ,
2    let  $defaults(i) = \{subst(U, iMap) | U \in iASDefs\}$ 
3    let  $premises(i) = \{subst(U, iMap) | U \in iASPrems\}$ 
4    let  $supports(i) = \{subst(U, iMap) | U \in iASSupps\}$ 
5    let  $attacks(i) = \{subst(U, iMap) | U \in iASAtts\}$ 
6    let  $undercuts(i) = \{subst(U, iMap) | U \in iASUCuts\}$ 
7
8  Let the set of arguments  $A = P \cup I \cup \{\eta\}$ 
9
10  $\forall p \in P, R_e = R_e \cup (\{\eta\}, p)$ 
11
12 if  $\exists i \in I, p \subset P$  such that  $p = premises(i)$ 
13    $R_e = R_e \cup (p, i)$ 
14 if  $\exists i \in I, p \subset P$  such that  $p = defaults(i)$ 
15    $R_a = R_a \cup (p, i)$ 
16  $\forall i \in I, A = A \cup supports(i)$ 
17  $\forall i \in I, \forall x \in supports(i), R_e = R_e \cup (i, x)$ 
18  $\forall i \in I, \forall x \in attacks(i), \text{ if } x \in A, R_a = R_a \cup (i, x)$ 
19  $\forall i \in I, \text{ if } \exists j \in I, x \in undercuts(i)$ 
20   where  $subst(x, j), R_a = R_a \cup (i, j)$ 

```

Figure 2. The algorithm used to convert a set of argument schemes and predicates to an evidential argument system.

An argument framework may have multiple evidential preferred extensions, each representing a set of arguments that a credulous reasoner would find consistent. Only one grounded extension exists, representing the arguments a sceptical reasoner should agree with.

The evidential argument system above treats arguments as abstract entities. We must therefore map between our system, which uses predicates and argument schemes, and the abstract arguments found in an evidential argument system, as shown in the algorithm of Figure 2. This mapping allows us to determine which predicates and argument schemes, are, in a sense consistent, by evaluating the appropriate extension over the resultant evidential argument system.

Within the algorithm, lines 1 to 7 define syntactic shortcuts used in the rest of the algorithm, with line 8 defining the initial set of arguments, made up of the predicates, η , and the instantiated argument schemes. While predicates and instantiated argument schemes have different types, we do not differentiate between them at the abstract level in which evidential argumentation systems operate. The remainder of the algorithm populates the attacks and support relations according to intuitive rules; lines 12 and 13, for example, link premises to argument schemes, while lines 14 and 15 cause the presence of a default to attack an argument scheme. The supported conclusions of an instantiated argument scheme, as specified in lines 16 and 17, must be added to the set of arguments, and the appropriate support relations must also be instantiated. Line 18 instantiates attacks between an argument scheme and its conclusions, while the final line creates attacks between an instantiated argument scheme and the schemes it undercuts.

Running the algorithm results in a system containing more predicates and instantiated argument schemes than a rational reasoner would believe are justified. By computing the evidential argument system’s grounded or preferred extension, we may determine what information and argument schemes are in fact consistent.

4 Argument Schemes for Normative Reasoning

We are now in a position to describe a number of argument schemes that an agent may use when reasoning about norms. We only consider obligations and permissions, and provide a very simple representation of these norms. Our first argument scheme deals with violations. We represent obligations using the one place predicate $obliged(G)$. If G does not hold, we assume that the obligation is violated. This leads to the following argument scheme:

$$\langle \{\}, \{obliged(G), not(G)\}, \{violated(obliged(G))\}, \{\}, \{\} \rangle$$

When reasoning about obligations, an agent must reason about the state of the world if an obligation is honoured. We assume that an agent honours obligations by default, leading to the following argument scheme:

$$\langle \{not(ignored(obliged(G))), not(violated(obliged(G)))\}, \{obliged(G)\}, \{G\}, \{\}, \{\} \rangle$$

Obligations may be conditional; for example, I may have an obligation to pay someone money if I buy something from them. We model this conditional by using the Modus Ponens argument scheme:

$$\langle \{\}, \{A, implies(A, B)\}, \{B\}, \{\}, \{\} \rangle$$

(Our use of Modus Ponens, together with the introduction of the $violation(\dots)$ predicate allows us to easily represent contrary to duty obligations.)

Now, permissions undercut obligations. That is, permissions explicitly allow us to ignore an obligation, by preventing the obligation argument scheme from coming into force.

$$\langle \{\}, \{permission(G)\}, \{\}, \{\}, \{obliged(not(G))\} \rangle$$

$$\langle \{\}, \{permission(not(G))\}, \{\}, \{\}, \{obliged(G)\} \rangle$$

Finally, we assume that an agent values some obligations more than others. This is captured, in the agent’s internal reasoning, via a “higher priority” argument scheme. This argument scheme undercuts the effects of the application of a norm, i.e. an agent reasoning that one norm is higher priority than another will not attempt to honour the lower priority norm. The defeasible nature of argument means that if the higher priority norm is itself successfully attacked (i.e. is not admissible), the lower priority norm will be reinstated.

$$\langle \{\}, \{obliged(A), higherPriority(obliged(A), obliged(B))\}, \{\}, \{\}, \{(\{not(violated(obliged(G)))\}, \{obliged(G)\}, \{G\}), \{\}, \{\} \rangle, \{A, G\}\} \rangle$$

5 Argument Schemes and Normative Conflict

We have now presented a procedure for an agent to determine which portions of an argument system are admissible, and proposed a number of general purpose and norm related argument schemes. However, we have not yet described how an agent may make use of these schemes in deciding which action to take.

The agent needs to perform inference, determining, from its knowledge base and argument schemes, which predicates hold. One further complication appears as the predicates may be influenced by those norms that the agent is willing to violate. As specified earlier, we assume that an agent has a knowledge base KB containing those

```

01 Given a set of predicates  $KB$ 
02 Given a set of argument schemes  $ASKB$ 
03  $i = 0$ 
04  $CS_0 = (A, R_a, R_e)$ 
05  $A = KB, R_a = R_e = \{\}$ 
06
07  $\forall p \in KB, R_e = R_e \cup (\{\eta\}, p)$ 
08 repeat
09    $i++$ 
10    $CS_i = CS_{i-1}$ 
11    $\forall AS = \langle ASDefaults, ASPremises,$ 
12      $ASSupports, ASAttacks, ASUndercuts \rangle \in ASKB$ 
13     if  $subst(ASPremises, M)$ 
14       where  $M \subset A$  and  $CS_i = (A, R_a, R_e),$ 
15        $CS_i = CS_i \cup \langle AS, M \rangle \cup subst(ASSupports, M)$ 
16 until  $CS_i = CS_{i-1}$ 
17 return  $CS_i$ 

```

Figure 3. The algorithm used by the agent to infer a knowledge base given an initial knowledge base KB and some initial argument schemes $ASKB$.

predicates that it believes hold in the environment, and that it is aware of a set of argument schemes, stored in $ASKB$, so that we can create a new knowledge base CS consisting of the agent’s original knowledge base, together with any inferences it may draw from its argument schemes. We may then calculate what may be inferred from CS , add it to CS , and repeat the process until no more inferences may be drawn. This is shown formally in Figure 3.

Note that the final CS_i will contain many instantiated argument schemes, and predicates, that may not be admissible. To determine what is admissible, we run the algorithm shown in Figure 2 over CS_i , starting at line 12, as we already have an argument framework. We may then compute the preferred extension to determine which predicates, and instantiated argument schemes, are actually admissible⁵. The presence of more than one preferred extension indicates that a normative conflict may exist. In this situation, the agent must explicitly add predicates of the form $not(honoured(O))$ (and possibly add additional predicates to reflect additional actions), where O is an obligation found in the agent’s knowledge base, and rerun the reasoning algorithm, until only a single preferred extension exists.

By attempting to honour, or not honour different obligations, suppositional reasoning may take place. If the agent associates utility gains and losses with various predicates, it may determine what norms to ignore and honour in such a way as to maximise its utility.

6 Example

We illustrate the framework using a simple example. Before examining the agent’s reasoning process in detail, we describe the scenario informally: A janitor (the agent) has an obligation to clean the floor, but needs a mop to do so. The mop is behind an alarmed door saying “authorised personnel only”. Of course, being a janitor, he is authorised. However, the door is also alarmed, and the janitor is prohibited from setting off the alarm. However, since it is part of his job, cleaning the floor is more important than not setting off the alarm. Clearly,

⁵ Note that this is a rather brute force approach to computing what holds, a dialogue game based approach (as described in work such as [6]) may be more elegant and computationally efficient.

the agent should decide to open the door, even though it means setting off the alarm. We assume that the janitor agent has the following predicates in its knowledge base:

```

implies(dirtyFloor, obliged(cleanFloor)), dirtyFloor,
obliged(not(openCupboard)), permission(openCupboard),
implies(openCupboard, setOffAlarm), obliged(not(setOffAlarm))
higherPriority(obliged(cleanFloor), obliged(not(setOffAlarm)))

```

Apart from the argument schemes described above, we assume that the agent has access to a planner, allowing it to determine what needs to be done to achieve a goal. This is obviously a simplification, but is sufficient for the purposes of illustration.

In this case, the argument system has a single extension, containing all the predicates found in the agent’s knowledge base (as no conflicts exist between them), as well as the predicates

```

obliged(cleanFloor), cleanFloor, openCupboard, setOffAlarm

```

Figure 4 shows the resultant argument system. Here, if the “higher priority” argument scheme was not admissible, two preferred extensions would have existed. The agent would then have had to decide which norm to violate to yield a single preferred extension.

7 Discussion and Future Work

The reasoning mechanism proposed in this paper is powerful, allowing for reasoning about a wide variety of situations to take place. However, this power comes at a cost; argument schemes must be defined for any situation that must be reasoned about. We proposed a number of argument schemes that may be used by the agent to reason about norms. This work is, however, preliminary and additional normative argument schemes dealing with issues such as power should also be present. Furthermore, our representation of norms is simple; we have ignored issues such as norm activation, maintenance and discharge (to model such issues requires the introduction of reasoning about temporal artifacts). Finding and representing additional norm-related argument schemes forms the backbone of our future work. We also intend to enhance our model, allowing for additional features such as argument schemes containing an arbitrary number of arguments.

Apart from argument schemes and predicates such as “higher priority”, we have no way of comparing the strength of attack or support. This concept is very useful, especially when dealing with uncertain concepts. Previous frameworks, such as the one described in [7], provide a powerful mechanism for assigning strength to arguments, but have difficulty handling loops. In the future, we aim investigate how argument strength may be integrated into the model.

Like AIF, the model described in Section 2 does not specify a structure for arguments, instead examining the interactions between predicates and argument schemes. Translating between our model and AIF is trivial, as concepts such as predicates map directly to I-Nodes, while argument schemes become R and C nodes. Unlike AIF, however, we are able to perform reasoning on our resulting structures. Recent work has suggested a more complex structure for argument schemes within AIF [9].

Our argument scheme structure, together with the presence of attack and support links within our abstract framework, enables us to easily model the most popular model of argument, namely the Toulmin model. Existing frameworks have long had difficulty supporting the richness of this model.

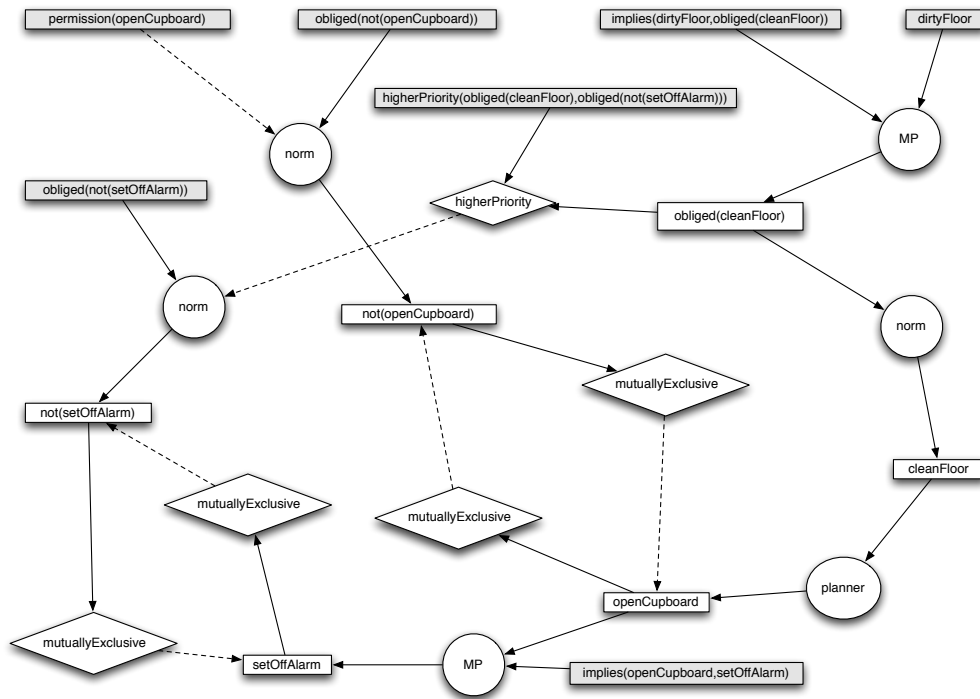


Figure 4. The evidential argument system representing the example; the shaded nodes represent those supported by η . Circular/oval nodes represent argument schemes intended to support certain conclusions, while diamond nodes represent attacking argument schemes. These are treated identically in the framework, but are differentiated for visual clarity. Solid arrows indicate support, while dashed arrows represent attacks.

8 Conclusions

In this paper we showed how an agent may use argumentation schemes and predicates to model its environment and normative state. By transforming this model into an evidential argument system, and computing the extension of this resulting argument system, the agent could perform different types of normative reasoning. This reasoning included detecting conflicts between norms, using additional knowledge from its knowledge base to overcome this conflict, and checking whether it has indeed fulfilled (or decided to ignore) all norms that affect it. Our framework was inspired by the way humans appear to reason when dealing with norms, and is able to easily handle normative conflict. Our approach is easily extensible, by adding extra domain specific argument schemes, the agent can cope with additional knowledge.

Acknowledgement: This work was undertaken as part of the CONTRACT project which is co-funded by the European Commission under the 6th Framework Programme for RTD with project number FP6-034418. Notwithstanding this fact, this paper and its content reflects only the authors' views. The European Commission is not responsible for its contents, nor liable for the possible effects of any use of the information contained therein.

REFERENCES

- [1] Trevor Bench-Capon, 'Value based argumentation frameworks', in *Proceedings of the 9th International Workshop on Nonmonotonic Reasoning*, pp. 444–453, Toulouse, France, (2002).
- [2] Claudette Cayrol and Marie-Christine Lagasque-Schiex, 'On the acceptability of arguments in bipolar argumentation frameworks', in *Proceedings of the Eighth European Conference on Symbolic and Quantitative Ap-*

- proaches to Reasoning With Uncertainty*, volume 3571 of *LNAI*, pp. 378–389. Springer-Verlag, (2005).
- [3] Carlos Chesñevar, Jarred McGinnis, Sanjay Modgil, Iyad Rahwan, Chris Reed, Guillermo Simari, Matthew South, Gerard Vreeswijk, and Steven Willmott, 'Towards an argument interchange format', *Knowl. Eng. Rev.*, **21**(4), 293–316, (2006).
- [4] Phan Minh Dung, 'On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games', *Artificial Intelligence*, **77**(2), 321–357, (1995).
- [5] Søren Holbech Nielsen and Simon Parsons, 'A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments', in *Proceedings of the Third International Workshop on Argumentation in Multi-Agent Systems*, pp. 7–19, Hakodate, Japan, (2006).
- [6] Nir Oren, *An Argumentation Framework Supporting Evidential Reasoning with Applications to Contract Monitoring*, Phd thesis, University of Aberdeen, Aberdeen, Scotland, 2007.
- [7] Nir Oren, Timothy J. Norman, and Alun Preece, 'Subjective logic and arguing with evidence', *Artificial Intelligence Journal*, **171**(10–15), 838–854, (2007).
- [8] John L. Pollock, *Cognitive Carpentry*, Bradford/MIT Press, 1995.
- [9] Iyad Rahwan, Fouad Zablith, and Chris Reed, 'Laying the foundations for a world wide argument web', *Artif. Intell.*, **171**(10–15), 897–921, (2007).
- [10] Douglas N. Walton, *Argumentation Schemes for Presumptive Reasoning*, Erlbaum, 1996.