Argument and Norms

Nir Oren
n.oren@abdn.ac.uk
Abstract: This paper describes a framework for practical reasoning in the presence of norms. We describe a formal normative model constructed using Action-based Alternating Transition Systems (AATSs). This model is able to represent the obligations and prohibitions present in a multi-agent system, as well as their violation, together with permissions which are used to derogate the former. Our model also captures the notion of a goal. Inspired by Atkinson’s scheme for practical reasoning, we then show how the possible executions of the system can be described through argumentation schemes and critical questions, allowing us to determine if sufficient information has been provided in order to perform practical reasoning, and to explain why specific sequences of actions were executed by agents within the system.

Keywords: Argumentation, Norms, Argument Schemes

1 Introduction

The violation of a norm, as expressed through obligations, permissions and prohibitions, can result in sanctions being imposed on the agent. Since such sanctions are undesirable, the agent will attempt to comply with its norms while pursuing goals. Norms therefore provide some control over the behaviour of an agent, imposing soft constraints upon it. When pursuing some goal, the agent must take its norms into account. If we assume some notion of rationality, the agent can violate a norm if the reward in doing so is greater than complying with the norm. Therefore, when performing practical reasoning, an agent must weigh up the penalties (and rewards) involved in violating (or adhering to) norms against the rewards provided by achieving its goals.

Now while practical reasoning frameworks taking norms into account have been previously proposed (e.g. [3]), explaining the decision processes taken by agents when acting in such a system, particularly to non-experts, is a difficult task. The aim of this paper is to investigate how argumentation can contribute to such explanations.

Decision and game theory provide processes whereby a rational agent (i.e. one that attempts to maximise some utility, or reach a most preferred state) can identify an optimal sequence of actions. We claim that in the practical reasoning domain, such processes can be more easily understood through argument schemes, with the evaluation of all instantiations of the scheme together with the relevant associated critical questions forming an argumentation framework. The evaluation of this argument framework will then identify the appropriate action(s) to pursue.

In this paper we propose a semantics for norms and goals that can be used to trace the execution of a system. The forward simulation of a normative system then forms the core of the practical reasoning process. Building on this formal system, we introduce an argument scheme together with the appropriate critical questions, which forms the heart of our approach.
In the next section we describe our formal model in detail. Following this, Section 3 introduces the argument scheme and maps it to our formal model. An example of the approach is provided in Section 4, and we discuss related and future work in Section 5, before concluding the paper in Section 6.

2 The Model

In this section we describe our formal model, which is based on action-based alternating transition systems (AATS) [9]. Such AATSs are intended to encode all possible evolutions of a system due to the actions of all agents within it, representing the various states through which the system can pass through by means of a branching time tree structure. Since this can be seen as a Kripke system, we can then describe a semantics over the system by means of a branching time logic. After introducing the basic concepts of AATSs, we detail how goals and norms, as well as more complex concepts such as violations and the derogation of obligations can be specified.

2.1 Semantics

Definition 1 (AATS, [9]) An Action-based alternating transition system (AATS) is a tuple of the form

\[ S = (Q, q_0, Ag, Ac_1, \ldots Ac_n, \rho, \tau, \Phi, \pi) \]

Where

- \( Q \) is a finite non-empty set of states.
- \( q_0 \in Q \) is the initial state.
- \( Ag = \{1, \ldots, n\} \) is a finite non-empty set of agents.
- \( Ac_i \), with \( 1 \leq i \leq n \), is a finite and non-empty set of actions for each agent, where actions for different agents do not overlap.
- \( \rho : Ac_i \rightarrow 2^Q \) is an action precondition function which identifies the set of states from which some action \( \alpha \in Ac_i \) can be executed.
- \( \tau : Q \times J_{Ag} \rightarrow Q \) where \( J_{Ag} = \prod_{i \in Ag} Ac_i \), is the system transition function identifying the state that results from executing a set of actions from within \( J_{Ag} \) in some state.
- \( \Phi \) is a finite and non-empty set of atomic propositions
- \( \pi : Q \rightarrow 2^{\Phi} \) is the interpretation function which identifies the set of propositions satisfied in each state.

Following [9], we define a computation (also referred to as a path) to be an infinite sequence of states \( \lambda = q_0, q_1, \ldots \). We index a state within a path using array notation. Thus, the first element of path \( \lambda \) can be referenced via \( \lambda[0] \), while a sub-path of the path starting at the second element and consisting of the remainder of the path is written \( \lambda[1, \infty] \). We label the set of all possible paths \( \Lambda \).

An AATS encodes the possible states of the world that result from executing actions, and can be viewed as a Kripke structure via the transition function \( \tau \). We can then represent the AATS through CTL* [4], allowing us to refer to both single paths and groups of paths in...
the structure. We define the semantics in two stages, first defining state formulae, following which we describe path formulae. Note that the syntax of the logic emerges directly from the semantics, and therefore, due to space constraints, is not detailed.

**Definition 2 (State Formulae)** State formulae are evaluated with respect to an AATS $S$ and a state $q \in Q$:

- $S, q \models \top$
- $S, q \not\models \bot$
- $S, q \models p$ iff $p \in \pi(q)$
- $S, q \models \neg \psi$ iff $S, q \not\models \psi$
- $S, q \models \psi \lor \phi$ iff $S, q \models \psi$ or $S, q \models \phi$
- $S, q \models A \psi$ iff $S, \gamma \models \psi$ for all paths where $\lambda[0] = q$
- $S, q \models E \psi$ iff $S, \gamma \models \psi$ for some path where $\lambda[0] = q$

Now, a strategy for an agent $i$ identifies the action that the agent should take in any state, and is therefore represented as a function $\sigma_i : Q \rightarrow Ac_i$. Similarly, a strategy profile for a group of agents $G = \{a_1, \ldots, a_k\}$ where $G \subseteq Ac$ is a tuple of strategies $\Sigma_G = \langle \sigma_1, \ldots, \sigma_k \rangle$. Given a state $q$ and a strategy profile $\sigma_G$, we define $\text{comp}(\sigma_G, q)$ to be the set of possible computations that exist starting at state $q$ and that occur when the group follows their strategy profile. This set of computations forms a tree of states, and $\lambda \in \text{comp}(\sigma_G, q)$ then identifies a path through the tree. From this, we define the following path formulae:

**Definition 3 (Path Formulae)** Path formulae are evaluated with respect to an AATS $S$ and a path $\lambda \in \text{comp}(\sigma_g, q)$:

- $S, \lambda \models \psi$ iff $S, \lambda[0] \models \psi$ where $\psi$ is a state formula.
- $S, \lambda \models \neg \psi$ iff $S, \lambda \not\models \psi$
- $S, \lambda \models \psi \lor \phi$ iff $S, \lambda \models \psi$ or $S, \lambda \models \phi$
- $S, \lambda \models \bigcirc \psi$ iff $S, \lambda[1, \infty] \models \psi$
- $S, \lambda \models \Diamond \psi$ iff $\exists u \in N$ s.t. $S, \lambda[u, \infty] \models \psi$
- $S, \lambda \models \Box \psi$ iff $\forall v \in N$ s.t. $S, \lambda[v, \infty] \models \psi$ and $\forall v$ s.t. $0 \leq v < u$, $S, \lambda[v, \infty] \models \psi$ and $\forall v$ s.t. $0 \leq v < u$, $S, \lambda[v, \infty] \models \psi$
- $S, \lambda \models \chi \triangleright \psi$ iff $\exists v \in N$ s.t. $S, \lambda[v, \infty] \models \psi$ and $\forall v$ s.t. $0 \leq v < u$, $S, \lambda[v, \infty] \models \psi$
- $S, \lambda \models \chi \triangleright \psi$ iff $\exists v \in N$ s.t. $S, \lambda[v, \infty] \models \psi$ and $\forall v$ s.t. $0 \leq v < u$, $S, \lambda[v, \infty] \models \psi$
- $S, \lambda \models \chi \triangleright \psi$ iff $\exists v \in N$ s.t. $S, \lambda[v, \infty] \models \psi$ and $\forall v$ s.t. $0 \leq v < u$, $S, \lambda[v, \infty] \models \psi$
- $S, \lambda \models \chi \triangleright \psi$ iff $\exists v \in N$ s.t. $S, \lambda[v, \infty] \models \psi$ and $\forall v$ s.t. $0 \leq v < u$, $S, \lambda[v, \infty] \models \psi$

Note that state formulae refer to states, and their semantics therefore refer only to a single state within a path, even in the case when the state operator then refers to a path formula (c.f. the $A$ and $E$ operators). Path formulae always refer to entire, infinite paths which begin at some state (e.g. the next state in the case of the $\bigcirc$ operator).

These semantics capture the evolution of a system over time due to agent actions. However, it says nothing about why one strategy should be followed over another in order to effect certain actions and therefore lead to certain states. To capture this notion we define a relation over paths, written $\triangleright_\phi$ to represent the preferences of some group of agents $g$’s of one group of paths over another. This group of paths is specified by means of a path formula. Thus, for example, $\Diamond a \triangleright_\alpha \Diamond \neg a$ captures the preference of agent $\alpha$ for those paths in which $a$ is eventually true over those paths where it is eventually false. When dealing with a single agent, or referring to a group by a label, we will often write $\triangleright_\alpha$ instead of $\triangleright_\{\alpha\}$. When the risk of confusion is low, we will compare two specific paths using the $\triangleright_\phi$ operator. Therefore, given two paths $\lambda$ and $\lambda'$, we write $\lambda \triangleright_\phi \lambda'$ if there is some pair of path formulae $\chi, \chi'$ such that $\lambda$ satisfies $\chi$ and $\lambda'$ satisfies $\chi'$ and $\chi \triangleright_\phi \chi'$. Finally, we write $\lambda \triangleright_\phi \lambda'$ to represent the case when $\lambda \triangleright_\phi \lambda'$ and $\lambda' \not\triangleright_\phi \lambda$, and abbreviate the situation where both $\lambda \triangleright_\phi \lambda'$ and $\lambda' \not\triangleright_\phi \lambda$ hold as $\lambda \sim_\phi \lambda'$.

Now a question arises as to the origin and form of the preference relation, and we propose that the agent’s goals, together with the norms found in the system constrain (but do not fully
specify) it. For example, a goal should imply that those paths in which the goal is achieved are preferred to those paths where it is not, but such a goal does not impose any preference ordering between those paths in which the goal is achieved (or indeed between those paths where it is not). We begin a more detailed exploration of the preference relationship by examining goals more closely.

### 2.2 Goals

Goals identify states of affairs in the world that an agent prefers (and should be able to bring about in part due to their action, but we do not formally impose this requirement). In other words, when undertaking practical reasoning, agents prefer those actions forming paths wherein their goals are achieved to those where they are not. We therefore represent goals through path formulae, identifying the state of affairs that must exist for a goal to be considered as met or satisfied.

Work such as [5] has identified two overarching types of goals, namely achievement goals and maintenance goals. The former identifies a state of affairs that must hold at some point in time, while the latter requires some state of affairs to be maintained until some deadline. Both of these goals can be easily represented in our logic. For example, a goal to achieve some proposition $x$ before a deadline $d$ can be written as $\neg d U x$, while a maintenance obligation requiring $x$ to be the case until deadline $d$ can be written as $(\neg d \land x) U d$. Note that open ended goals (e.g. a goal to simply achieve $x$, written $\Diamond x$) are also possible. In fact, all of Allen’s interval operators [1] can be captured in our logic.

We begin by assuming that goals are not conditional (i.e. a goal cannot be of the form $X \rightarrow \gamma$ where $X$ is some logical sentence and $\gamma$ is a goal). In such a situation, a maintenance goal must be adhered to from the current point in time until its deadline in order to be seen as achieved. Since achieving a goal $\gamma$ is preferred by the agent over not achieving the goal, we identify a preference ordering over possible paths, expressed by the simple rule

$$\Diamond \gamma \triangleright^o \Box \neg \gamma$$

That is, paths in which the goal is (eventually) achieved are preferred by the agent to those in which the goal is never achieved.

### 2.3 Norms

Norms within a system represent obligations, prohibitions and permissions imposed on, or provided to, entities within a society or group. Obligations and prohibitions (respectively) identify the states of affairs that a target must ensure do (or do not) occur. If these states of affairs do not/do occur, then the norm is violated. We treat permissions as exceptions to obligations and prohibitions: in the case of an obligation, if a state of affairs is ordinarily obliged, but a permission not to achieve the state exists, then even if the state of affairs is not achieved, no violation occurs.

Now we view norms primarily as social constructs. That is, an obligation (for example) specifies who should behave in some way (i.e. it has a set of target agents), and also identifies which agent — or set of agents — desires that this behaviour occur. The latter form the norm’s creditors (c.f. [8]).

Following this perspective, we view a norm as expressing a preference over a state of affairs for its creditors rather than its target. That is, a creditor prefers those situations in which a

---

1 The semantics of $U$ are existential, that is, we need to ensure that the deadline does not occur before it does.

2 The inclusion of conditional goals via material implication would require paths in which the condition does not occur but the goal is satisfied to be less preferred to paths where the conditional does not occur.
norm is not violated to one where it is. Now this implies that a norm, in isolation, has no direct effect on its target's behaviour. Instead, we claim that such behaviour regulation stems from two sources. First, the violation of a norm could (via contrary-to-duties) permit a sanction to be imposed on the violator. Second, social ties could mean that a norm's target takes the norm creditor's preferences into account (e.g. I may fulfil my obligations to my friends because I care about their feelings rather than any threat of sanctions). Note however that in our argument framework, we merge all individual agent preferences into a global preference, limiting the effects of this approach; investigating a more “local” view of preferences forms part of our future work.

Now, obligations identify states of affairs that should be achieved by the target of the obligation. Obligations are social constructs, imposed by some group (the creditor) on the target. Furthermore, if an obligation is not fulfilled, then the creditor could potentially sanction the obligation’s target. An obligation therefore encodes two concepts, namely the preference by the creditor for paths wherein the obligation is fulfilled over those where it is not. Second, if an obligation is not fulfilled, then a record must be kept that it has been violated.

We consider two distinct types of obligations, namely achievement obligations, which require the target to see to it that some state of affairs holds at some point before some deadline occurs, and maintenance obligations, which require the target to ensure that the state of affairs holds at all points before the deadline. Before formally defining obligations, we must examine the notion of a permission, which we consider as an exception to an obligation.

Permission acts as an exception to an obligation (or a prohibition). In other words, given an obligation to achieve some state of affairs, and a permission not to achieve it, not achieving this state of affairs will not result in a violation. Now, as discussed previously, we model prohibitions through obligations, and therefore concentrate on the interactions between permissions and obligations. Like other modalities, a permission is given by some creditor to a target, and affects the creditor’s concept of a violation. Similarly, permissions identify some (permitted) state of affairs, and a deadline.

Clearly, interpreting a permission in this way makes little sense without an obligation or prohibition being present, and we therefore encode permissions through the presence of a permission predicate\(^3\) which prevents a violation from occurring. We write this permission predicate as \(perm(a, g, x)\) where \(a\) is the target of the permission, \(g\) the creditor and \(x\) the permitted state of affairs. Together, these parameters uniquely identify a permission. Formally, a permission is then defined through the following equivalence:

\[
P^p_a(x|d) \equiv Aperm(a, g, x) U d
\]

Since we must ensure that the permission predicate does not hold when no permission is in force, we must require the following axiom in the system:

\[
A □ (¬P^p_a(x|d) \rightarrow ¬perm(a, g, x))
\]

Obligations identify states of affairs that should hold, and a failure to abide by the requirements of an obligation leads to a violation. We encode such a violation through a predicate, in a manner similar to the permission predicate. In other words, the predicate \(viol(a, g, x, d)\) represents a violation by the target \(a\) of the obligation, with respect to a creditor \(g\), to see to it that state of affairs \(x\) was the case with respect to a deadline \(d\).

An achievement obligation requiring the target to ensure that some state of affairs holds before a deadline is represented as follows:

\[
O^a_a(x|d) \equiv A(¬viol(a, g, x, d) \land ¬d \land ¬x) U (((¬x \land d \land ¬perm(a, g, ¬x) \land viol(a, g, x, d)) \lor (¬x \land d \land perm(a, g, ¬x) \land ¬viol(a, g, x, d))) \lor (x \land ¬viol(a, g, x, d)))
\]

\(^3\)This predicate can be trivially translated to a proposition.
That is, an obligation on target \( a \) from creditor \( g \) to see to it that \( x \) is the case before deadline \( d \) requires the following conditions to be met on all possible paths:

1. Before either the deadline or \( x \) occurs, the obligation is not considered violated.

2. If the deadline occurs and \( x \) is not the case, then if there is no permission to maintain \( \neg x \), then a violation is recorded. Alternatively, if such a permission exists, then no violation is recorded.

3. Finally, if \( x \) is achieved (before the deadline), then no violation is recorded.

Therefore, our encoding of an obligation essentially states that if an obligation is in force, it is only violated if the deadline is reached without the desired state of affairs being achieved, assuming that no permission exists allowing the obligation to be ignored. However, nothing in this definition prevents a violation from existing in a state of affairs without an associated obligation. We therefore require that the following axiom hold:

\[
\mathcal{A} \Box (\neg O^g_x(m : d) \rightarrow \neg viol(a, g, x, d))
\]

Maintenance obligations requires that a state of affairs be maintained until some deadline\(^4\), and are defined as follows:

\[
O^g_x(m : d) \equiv \mathcal{A} ((\neg x \land \neg d \land (\neg perm(a, g, \neg x) \land viol(a, g, m, d)) \lor (perm(a, g, \neg x) \land \neg viol(a, g, m, d))) \lor (x \land \neg d)) \land d
\]

In other words, before the deadline, either \( x \) is maintained, or \( x \) is not maintained, in which case the obligation is violated if an associated permission does not exist.

The requirement for the lack of a violation, as stated above, is repeated for maintenance obligations: \( \mathcal{A} \Box (\neg O^g_x(x : d) \rightarrow \neg viol(a, g, x, d)) \)

In discussing obligations so far, we have identified the situations in which they are violated. Detecting these situations allows for the modelling of contrary to duty obligations, which come into force when a violation occurs. Such contrary to duties are a form of conditional obligation, which comes into force only when some state of affairs holds in the environment, and generally, such conditionals can be represented via an implication relation, for example, \( viol(a, g, x, d) \rightarrow O^g_x(x' | d') \).

We now turn our attention to the second aspect of obligations, namely their interactions with preferences over paths through the system. Informally, the presence of an obligation or prohibition imposed by some creditor leads to that creditor preferring those paths through the system where the obligation is complied with (i.e. not violated) over those where it is violated.

This leads to the following rule within our system:

\[
\Box (\neg viol(a, g, x, d)) > g \Diamond viol(a, g, x, d)
\]

Note that we do not prefer fewer violations over more violations, as other preferences, for example regarding the interval length of a violation, could affect the preference ordering.

In this work we consider only achievement prohibitions, that is, prohibitions on seeing to it that a state of affairs holds (until the prohibition’s deadline occurs). Such a prohibition can in fact be modelled as a maintenance obligation: \( F^g_x(m : d) \equiv O^g_x(\neg x | d) \)

\(^4\)We assume that this maintenance requirement comes into force with the obligation, ignoring obligations of the form “maintain x between 5pm and 8pm tomorrow”.

6
3 Argumentation Scheme

Our formal model contains two distinct aspects. The first aspect consists of the AATS, which identifies all possible evolutions of the system, while the second aspect is associated with the preferences over paths (i.e. sequence of actions). From the practical reasoning point of view, each path through the AATS—which identifies one possible run within the system—can be justified, and represents a single instance of the argument scheme.

**AS1:** In situation $S$, joint actions $A_1, \ldots, A_n$ should be executed.

This argument scheme is associated with two critical questions:

CQ1-1 Does some other sequence of actions exist that can be executed?

CQ1-2 Is there a more preferred sequence of actions to this one?

Now in order to justify why one sequence of actions should be followed over another, we must show that it is the most preferred action sequence, and explain why this is the case. In order to do this, we now introduce additional argument schemes which are associated with preferences over paths.

**AS2:** The sequence of joint actions $A_1, \ldots, A_n$ is preferred over $A'_1, \ldots, A'_n$ as the former achieves a goal which the latter does not. Critical questions here are as follows:

CQ2-1 Is there some other sequence of actions which achieves a more preferred goal than the one achieved by this action sequence?

CQ2-2 Does the sequence of actions lead to the violation of a norm?

We can associate two related argument schemes with obligations and prohibitions:

**AS3:** The sequence of actions $A_1, \ldots, A_n$ should be less preferred than sequence $A'_1, \ldots, A'_n$ as the former violates a norm while the latter does not.

CQ3-1 Is the goal resulting from the sequence of actions more preferred than the violation?

CQ3-2 Does the violation resulting from this norm result in some other, more preferred norm not being violated?

CQ3-3 Is there a permission that derogates the violation?

**AS4:** There is a permission that derogates the violation of an obligation.

Finally, we can identify several simple argument schemes that allow an agent to associate preferences between different goals and norms:

**AS5:** Agent $\alpha$ prefers goal $g$ over goal $g'$

**AS6:** Agent $\alpha$ prefers achieving goal $g$ to not violating $n$

**AS7:** Agent $\alpha$ prefers not achieving goal $g$ to violating $n$

**AS8:** Agent $\alpha$ prefers violating $n$ to violating $n'$

We now provide a formalisation of the argument schemes and critical questions based on our AATS semantics. Below, we speak about action sequences which are equivalent to paths through the AATS. We label our AATS $S$, and write $\overline{S}$ for the logical closure of $S$.

**AS1:** Given an initial state $q_0 \in Q$ and a sequence of joint actions $j_1, \ldots, j_n$ such that for all $i = 1, \ldots, n, \tau(q_{i-1}, j_i) = q_i$, this joint action sequence should be executed.

**AS2:** Given an initial state $q_0 \in Q$ and a preference relation over paths $\succeq = \bigcup_a \succeq^a$ and two paths $\lambda, \lambda'$ where $\lambda$ is the path obtained from the sequence of joint actions $j_1, \ldots, j_n$ such that for all $i = 1, \ldots, n, \tau(q_{i-1}, j_i) = q_i$, and $\lambda'$ is the path obtained from the sequence of joint actions $j'_1, \ldots, j'_n$ where $i' = 1, \ldots, n, \tau(q_{i'-1}, j'_{i'}) = q_{i'}$ there is a goal $g$ such that $S, \lambda \models g$ and $S, \lambda' \not\models g$. 

**AS3**: Given an initial state $q_0 \in Q$ and two paths, $\lambda, \lambda'$ where $\lambda$ is the path obtained from the sequence of joint actions $j_1, \ldots, j_n$ such that for all $i = 1 \ldots n$, $\tau(q_{i-1}, j_i) = q_i$, and $\lambda'$ is the path obtained from the sequence of joint actions $j'_1, \ldots, j'_n$ where $i' = 1 \ldots n$, $\tau(q'_{i-1}, j'_{i'}) = q'_i$, $S, \lambda \models O^g(a, x, d)$, and $S, \lambda' \not\models O^g(a, x, d)$.

**AS4**: Given an initial state $q_0 \in Q$ and a path $\lambda$ obtained from the sequence of joint actions $j_1, \ldots, j_n$ such that for all $i = 1 \ldots n$, $\tau(q_{i-1}, j_i) = q_i$, and $\lambda'$ is the path obtained from the sequence of joint actions $j'_1, \ldots, j'_{n'}$ where $i' = 1 \ldots n'$, $\tau(q'_{i'-1}, j'_{i'}) = q'_{i'}$, we define these argument schemes as follows:

- **AS5**: $S, \lambda \models g$ and $S, \lambda' \not\models g'$ and $g \geq^\alpha g'$
- **AS6**: $g \geq^\alpha \neg \text{viol}(\alpha, h, x, d)$ for some $h, x, d$
- **AS7**: $\neg g \geq^\alpha \text{viol}(\alpha, h, x, d)$ for some $h, x, d$
- **AS8**: $\text{viol}(\alpha, h, x, d) \geq^\alpha \text{viol}(\alpha, i, y, e)$ for some $h, x, d, i, y, e$
- **AS9**: $A \geq^\alpha B$ where $A, B$ are formulae in our language.

Now let us turn our attention to the critical questions, using the same definitions as above.

**CQ1-1**: There is a sequence of joint actions $j'_1, \ldots, j'_n$ such that for some $i \in 1 \ldots n$, $j_i \neq j'_i$.

**CQ1-2**: There is an instance of AS2 or AS3 whose path $\lambda$ is this instance of AS1.

**CQ2-1**: There an instance of AS5 whose less preferred goal is the one identified by this instantiation of AS2.

**CQ2-2**: There is an instantiation of AS7 which refers to some AS2 which has the same goal $g$ and AS3 with norm $n$.

**CQ3-1**: There is an instantiation of AS6 which refers to some AS2 which has the same goal $g$ and AS3 with norm $n$.

**CQ3-2**: There is an instantiation of AS8 which refers to two instances of AS3 with obligations or prohibitions $n, n'$.

**CQ3-3**: There is an instantiation of AS4 which refers to some instance of AS3 with obligation or prohibition $n$.

### 3.1 Instantiating the Framework

We instantiate the framework described above using Modgil’s extended argument frameworks [6] (EAF). Formally, an EAF is defined as follows:

**Definition 4 (Extended Argument Framework)** An EAF is a tuple $(\text{Args}, R, D)$ such that $\text{Args}$ is a set of arguments, $R \subseteq \text{Args} \times \text{Args}$, and $D \subseteq \text{Args} \times R$ subject to the constraint that if $(C, (A, B))$, $(C', (B, A)) \in D$, then $(C, C')$, $(C', C) \in R$.

Now, each instantiation of any of the argument schemes is associated with an argument within our EAF, and each critical question is associated with an attack. The constraint imposed on EAFs causes additional attacks to appear that are not described by the critical questions. We describe the process of EAF instantiation informally due to both space concerns and its simplicity.

CQ1-1 arises since only one sequence of actions can ultimately be executed, and results in symmetric attacks being inserted into $R$ between every pair of nodes instantiating AS1. CQ2-2
refers to preferences between actions and following [7], is captured via an attack from the node representing the argument to the appropriate attacking edge.

CQ2-1, CQ2-2, CQ3-1 and CQ3-2 capture preferences over goals and norms. That is, they are used to represent the fact that one goal (or norm) is preferred over some other goal (or norm) by entities in the system. All of these link the appropriate argument, as instantiated by AS5-8 via an attack, on the attack from the argument instantiated by the appropriate AS2 or AS3.

Finally, CQ3-3 encompasses the possibility of a violation being derogated by a permission, and in instantiated as an attack from AS8 to the appropriate AS3.

Given the above, CQ1-1 and CQ3-3 result in attacks added to \( R \), while the remaining critical questions result in attacks added to \( D \). Together with the attacks added by the constraint, these attacks between the arguments instantiated from the application of the argument schemes fully specify our EAF.

Given an EAF instantiated as above, all the preferred extensions of the EAF will contain a single argument from argument scheme AS1 for some specific action sequence to be executed iff this action sequence is most preferred by all agents in the system. This sequence of actions is the dominant strategy for all agents in the system.

4 Example

In this section, we provide a brief example of the framework in action. Due to space constraints, we do not present all details of the system in our example, but instead concentrate on the most important aspects of the system’s operation.

Consider two agents, \( \alpha \) and \( \beta \). \( \alpha \) can undertake two actions, namely to visit her ill mother in hospital (\( V \)), or go to work (\( W \)). \( \beta \), who is \( \alpha \)’s boss, has two possible actions, namely to fire \( \alpha \) (\( F \)), or not fire her (\( N \)). \( \alpha \) has two (conflicting) goals: to visit her mother (\( vm \)), and to keep her job (\( kj \)), while \( \beta \) would like to see some work done (\( wd \)), which can only occur if \( \alpha \) goes to work. Finally, \( \beta \) has an obligation to not fire \( \alpha \), but has permission to do so if she does not turn up to work.

The AATS for this example is shown in the top left of Figure 1, and instantiating the EAF results in the main graph of Figure 1. Within this graph, paths from the AATS are indicated through nodes containing the path number; preference information is encoded through the propositions true in the state (e.g. \( 1 > 3 \ kj \) indicate that path 1 is preferred to path 3 due to \( \alpha \)’s preference to keep her job; the permission to fire is indicated via the \( per \) node, and \( nvl \) identifies preference nodes instantiated through the prohibition on firing \( \alpha \). Dashed lines indicate attacks due to actions being mutually exclusive, while solid lines capture preference based attacks.

Evaluating the preferred extension of this EAF indicates that multiple actions are possible; for example, paths \{1,2\} are present in one of the extensions. This means that the system’s preferences are underspecified. Looking at the situation more closely, this occurs for several reasons. First, \( \alpha \) does not have any preferences encoded between going to the hospital or keeping her job; prioritising one of these (by adding attacks on edges between \( kj \) and \( vm \)) reduces the number of arguments for action in the extension, for example, if \( vm \) is preferred over \( kj \), only path 2 remains in the extension indicating that \( \alpha \) should visit her mother and keep her job.

This odd result arises because while \( \beta \) has permission to fire \( \alpha \) if she does not turn up to work, no preferences are expressed over whether \( \beta \) would prefer this situation to one where \( \alpha \) keeps her job. Adding an additional preferences over paths, through a new goal for \( \beta \) stating that either the work is done and \( \alpha \) keeps her job, or the work is not done and \( \alpha \) is fired, will
result in $\alpha$ losing her job if she visits her mother in hospital (path 4). Note that due to the permission, there is no need to then express another preference for $\beta$ between this goal and the norm on not firing $\alpha$; without the permission, such an additional preference would be necessary.

5 Discussion and Future Work

In practice, there are several ways of using the framework proposed here, each of which poses an avenue for future research. First, as done in the example above, a given AATS could be converted to an EAF and evaluated in order to identify whether sufficient preference information has been provided in order to reach a decision about a sequence of actions. The potential exponential growth in the number of arguments with respect to AATS size makes this approach practical for only small AATSs.

Second, a dialogue game could be formulated — and verified against an AATS — based on arguments and attacks instantiated from the argument schemes and critical questions. This would then involve agents arguing for why some course of action should be taken and providing their preferences for certain outcomes as appropriate.

Third, and perhaps most novel, an instantiated EAF could be used as the basis of a process to explain why some sequence of actions was followed given agents with some goals and norms.
A user could, for example, understand that an action was executed as while a norm was violated, the goal achieved was more important to the agents in the system than the violation.

Our work borrows several ideas from Atkinson’s argument scheme for practical reasoning based on values [2]. Atkinson’s approach puts both goals and values at the centre of the argumentation scheme, stating that “in situation $S$, action $A$ should be pursued in order to achieve goal $G$ while promoting values $V$”. This argument scheme is encoded through a VAF, which is used to represent the preferences of different audiences over values. Each argument within the VAF can be associated with several values, but an audience’s value ordering must be fully specified and consistent.

Now in the current work, preferences (which have a similar role to values in Atkinson’s work) are associated with different sequences of action due to the goals that these sequences achieve for the agents as well as the norms violated or complied with by the sequence. Given this, our AS-1 argument scheme is much simpler than Atkinson’s, stating that (by default) some sequence of actions should be executed, and requiring all possible sequences of actions to be mutually exclusive with each other. Deciding how to act then requires identifying the most preferred sequence of actions.

Our representation of preferences within an EAF is based on [7], which applied EAFs to VAFs. While there are many similarities between our instantiated EAF and the EAF based VAF approach, the requirement of VAFs to have a single consistent preference ordering makes them unsuitable for our needs; as shown in the example, we explicitly concern ourselves with detecting inconsistent preference orderings.

In one sense, the work presented here takes a global view of norms and actions. We consider joint actions, and require that all agents agree on a path. Such an approach ignores an important nuances of practical reasoning: agents may be force to pursue sub-optimal goals due to the actions of other agents. Thus, while our approach currently finds dominant strategies, it is unable to find other game theoretic solution concepts (e.g. Nash equilibria); we believe that capturing these additional solution concepts is critical, and are currently investigating how these concepts can be captured using our approach.

Another avenue of future work involves integrating our practical reasoning over norms with reasoning over values. This, in combination with the already present capability to reason over goals, should provide an end-to-end practical reasoning formalism.

6 Conclusions

In this paper we proposed a representation for norms built on top of an AATS. Using this representation we described how arguments over norms can be constructed, allowing for the detection of inconsistencies when performing practical reasoning, the explanation of why some action was taken, and making a decision about how to act in the presence of both goals and norms.

References


