Engineering norm-aware BDI agents

Michael Winikoff[1][0000−0002−5545−7003], Frank Dignum[2][0000−0002−5103−8127], Sebastian Rodriguez[3][0000−0002−0514−9221], and John Thangarajah[3][0000−0002−7699−6444]

[1] Victoria University of Wellington, New Zealand
michael.winikoff@vuw.ac.nz
[2] Umeå University, Umeå, Sweden
dignum@cs.umu.se
[3] RMIT University, Melbourne, Australia
{sebastian.rodriguez,john.thangarajah}@rmit.edu.au

Abstract. Norms are an important abstraction for multi-agent systems (MAS) because they specify *flexible constraints*. As constraints they help bring order to agent behaviour. However, being flexible, they can be violated, and thus do not over-constrain autonomous entities. In order to build MAS with cognitive agents that use norms we need to be able to engineer these agents to be *norm-aware*, i.e. being able to take norms into account in their reasoning. Such agents need the ability to decide to: (1) knowingly *violate* a norm, whilst understanding the consequences; (2) *conform* with a norm, which may require adapting its plans, e.g. adding extra steps; and (3) change their choice of plan for a given goal, due to consequences of a norm. Existing work on norm-aware agents does not cover all these cases. Additionally, to support adoption, an approach to engineering norm-aware agents should be usable with existing agent-oriented programming languages. This paper defines an approach, including a step-by-step process, to engineering norm-aware BDI (Belief-Desire-Intention) agents that handle these three cases and that can be used with off-the-shelf BDI agent languages.

1 Introduction

Over the past thirty years many people have argued that norms are important for MAS [6,2,11,19,18,15]. The main argument is that norms provide a useful abstraction for autonomous systems because they specify *flexible constraints*. They usually are obeyed and then simplify interactions and generate expected behaviour. However, because they can be violated, they also are not like a straightjacket that makes the MAS behave rigidly in situations where the norms are less useful or are having consequences that are unacceptable for one or more parties. E.g. during the COVID crisis people would go out despite a complete lockdown, because they felt they needed to see their family.

Once one accepts the importance of norms and their fundamental feature that they can be violated, it becomes important for agents to be able to reason when they can and should violate a norm and when they should adapt their behaviour to comply with it [20,21]. Thus agents need to be able to reason about norms, and their consequences and adapt their planning and behaviour accordingly. In particular, we need to consider

the following features/cases: (1) a norm has consequences, and in order to avoid violating it we need to do additional things; (2) a norm has consequences, but we choose to violate it (knowingly) regardless; (3) the consequences of a norm lead us to change our choice of plan to realise a given goal.

This makes adding norms to a MAS complex, since the norms influence the agents' planning process. Furthermore, if we want norms to become a mainstream feature of MAS platforms it is important that we can easily add norm-awareness (including the cases above) in existing Agent-Oriented Programming Languages (AOPLs) without having to modify the platform. Although there has been prior work on norm-aware agents, none of the approaches (see §1.1) cover these cases and are usable with existing (unmodified) BDI platforms.

This paper proposes an approach for engineering norm-aware cognitive agents, specifically using BDI (Belief-Desire-Intention) AOPLs. The core idea of our approach is to transform each norm into a goal-plan tree in a way that allows the agent to be norm-aware. We define a simple *process* for doing so (§3) and show how we can realise norm-aware BDI agents in Jason (§4).

Our paper uses the following running example: we have a goal to go to work, and can do this in four different ways: walking (if the distance is small), using a ride sharing service, driving ourselves (if we have access to a car), or using public transport ("PT", which involves the steps: go to the station, board the train, wait until one arrives at the destination, and then disembark). The use of public transport is subject to a norm that passengers must have a valid ticket when using public transport, or be subject to a fine.

In this context the sorts of reasoning that we want our norm-aware Jason agents to be able to do, if it chooses the PT option, include: (1) adapting its plan to add buying a ticket (if it decides to comply with the norm); (2) deciding to violate the norm *knowingly* by taking account of the consequences of doing so (the possible fine); and (3) when reasoning about what course of action to choose, take account of the consequences of violating the norm and of the additional action(s) required to conform with the norm.

1.1 Related Work

Meneguzzi & Luck [18] provides a way for BDI agents to deal with obligation norms by adding goals, and to deal with prohibition norms by *suppressing* plans. The approach is implemented by adding plans to an AgentSpeak(L) agent (extended with meta-reasoning facilities). There is no ability for an agent to make a decision to intentionally violate a norm.

Alechina *et al.* [2] presented a norm-aware BDI language, N-2APL. Their focus was on the case where an agent was unable to achieve all goals and respect all norms, and how to have a rational schedule that did the best it could. The focus is therefore on the case where goals need to be dropped (or norms violated), in particular in relation to *time*. Assumptions made include that there is a known priority over goals and sanctions, and that it is more important to do something with a certain priority, than to do any number of lower priority things. They also assume that the time to execute a plan is fixed and known in advance.

Lee *et al.* [15] extends the work of Alechina *et al.* [2] and defines N-Jason, which extends Jason with additional reasoning. Events and plans have deadlines and priorities

Paper(s)	Add Steps	Violate Norm	Select Plan	Off-the-shelf BDI?
[18]	✘	✘	*see text*	✘
N-2APL [2]	✘	✔	✘	✘
N-Jason [15]	✘	✔	✔	✘
NBDI [25,24]	✘	✔	✔	✘
ν-BDI [17]	✘	✔	✔	✘
JaCaMo [3]	✘	✔	✘	✘
Social Commitments [7,36]	✘	✘	✔	✔
This paper	✔	✔	✔	✔

Table 1. Features of prior work. Notation: "✔" = supported, "✘" = not supported.

(specified using Jason annotations). The reasoning mechanism is extended with additional steps to handle newly activated norms, and to schedule intentions, taking into account deadlines and norms. Similarly, NBDI [25,24] supports reasoning about goal selection, allowing for norm violation, but involves extending the implementation.

Meneguzzi *et al.* [17] propose ν-BDI. They give a logical formalisation of a framework that allows BDI reasoning to decide which norms to violate (assuming that utilities are known), and to select the best plans (for maximal utility).

JaCaMo [3] integrates Moise and Jason. It does not provide mechanisms for agent reasoning about norms, but rather only an integration that prompts the agent (with events) when it has e.g. a new obligation.

There is work on implementing social commitments using BDI languages [7,36]. While commitments can be seen as a form of norm, they typically do not include information on the sanctions associated with violating the commitment. Consequently, they do not support reasoning about whether to (intentionally) violate a commitment or not, taking the consequences into account.

Table 1 summarises prior work with respect to key desired features: the ability to support the *addition of steps* due to norms (e.g. buying a ticket to avoid a fine), the ability to intentionally *violate norms*, the ability to take norms into account when *selecting plans*, and whether it works with off-the-shelf BDI platforms.

2 Background

This section introduces required background: norms (§2.1), Goal-Plan agents (§2.2) and values, valuings and preferences (§2.3).

2.1 Norms

There are many different definitions of norms usually referring to differences in how the norms emerge and are maintained. In general one can distinguish *social norms* from *injunctive* or *legal* norms [10]. Social norms [14,5,27] emerge from repeated patterns of behaviour and over time become a kind of standard of behaviour. These norms are maintained by repeating behaviour according to the norm and they disappear when no-one behaves according to the norm or if the context has changed in a way that the norms

hardly ever are applicable anymore. These types of norms are most often used in agent based social simulations.

In contrast we have *injunctive* or *legal* norms [34,17,2,11] that are explicitly agreed upon norms that regulate the behaviour of a society or group within the society. When talking about norms in MAS we usually talk about this latter type of norms. The norms are explicitly given and engineered into the system. Therefore the agents do not need to have mechanisms to cope with emerging norms or norm maintenance. The agents only need to reason about how the norm should influence their behaviour.

In order to describe the norms we make use of the most common format introduced by Crawford & Ostrum [9] called ADICO. The abbreviation indicates the different elements that need to be specified for a norm. E.g. a norm such as: A passenger using public transport must have a valid ticket, is specified as:

A(ttributes): a person who is a passenger of public transport
D(eontic): must
a**I**m: have a valid ticket
C(ondition): when riding public transport
O(r else): pay a fine.

In this framework Deontic options are "must", "must not" and "may"[4] The Aim is either an action or a condition, and the norm is applicable if both the attributes and the condition hold. Although the "Or else" is written as an action to be performed it usually is a new norm. In the above example the passenger must pay a fine. This can then be described through another norm. This sequence ends if the punishment can be forced upon the violator.

2.2 Goal-Plan Agents

Goal-Plan agents (also called BDI agents), are defined in terms of a reasoning process that uses goals[5], plans, beliefs, and intentions. There are a number of BDI Agent Oriented Programming Languages (AOPLs), including Jason [4] and JACK [35].

Each agent has *goals* that it wishes to bring about, *beliefs* about the state of the world and itself, and a collection of *plans* that can be used. A plan has an indication of what goal it is relevant for, and a *context condition* that is evaluated when plans are being selected for a given goal instance to determine whether the plan is *applicable* at that point in time. Finally, each plan has a *plan body* that defines what the plan does. The plan body can include posting sub-goals, performing actions, modifying beliefs, and testing conditions.

A Goal Plan Tree (GPT) is a standard abstraction [30,31] that views an agent's collection of plans as an And-Or tree where goal nodes have as children ("Or") the plans that are relevant for that goal, and each plan node has its sub-goals as children ("And"). In our GPTs, for clarity, we also include actions as children of the plan where they appear.

[4] Our paper does not consider these, since they do not constrain behaviour.
[5] Sometimes referred to as desires, although the concept is subtly different.

2.3 Values, Valuings and Preferences

In order to reason about norms the agents have to reason about the consequences of following a norm or violating it. In the literature the two are often compared using the utilities of both. E.g. buying a ticket for PT costs a few dollars, while not buying a ticket lets me ride for free unless I get caught and need to pay a fine. However, it is known that people usually do not make these simple comparisons. We use more fundamental comparisons that indicate that following a norm might have some costs but also gives a sense of being a good citizen and not having to worry about the possible embarrassment of getting caught. The three key concepts we use are values, valuings, and preferences:

- **Values** are high-level general drivers for a certain type of behaviour favouring a type of state of the world, e.g. hedonistic, social good, conservative and openness to change [28].
- **Valuings** are specific properties of a particular option (e.g. fast, certain cost), defined by Malle [16] as things that "*directly indicate the positive or negative affect towards the action or its outcome*".
- **Preferences** apply to two (or more) options indicating which is preferred, based on their valuings (not all of which would be equally important in a given context), and the agent's values. For example, preferring to use public transport over driving for a particular trip.

We will follow [8] and use valuings as a way of capturing relevant (and more concrete than values) attributes for each option, grouping together a diverse set of criteria without having to aggregate them into a kind of utility. We then define preferences over the valuings. These preferences can take values into account where needed, for example, that in the context of selecting transport options, a strong value of social good can affect the preference between options.

Using valuings we can characterize the different transport options in the following way. We assume that cost(ticket) $<$ mediumCost and cost(fine) $<$ mediumCost. Note that the cost of the fine is an estimate, taking into account the actual fine but also the anticipated likelihood of being fined if the norm is violated (see §5.1 Case 2).

Option	**Valuing**
Walk	Slow, Sustainable, noCost
Rideshare	Fast, highCost
Drive	Fast, mediumCost (parking, petrol)
Use PT	medium speed, Sustainable
Violate norm	Cost(fine), reputation loss
Comply with norm	(public good) Cost(ticket)

3 Adding norms to BDI agents

Our overall process is the following (see Figure 1), starting with a goal-plan tree G, and one or more norms N_i.

1. Translate each relevant norm N_i into a corresponding Goal with GPT G_{N_i} (§3.1).

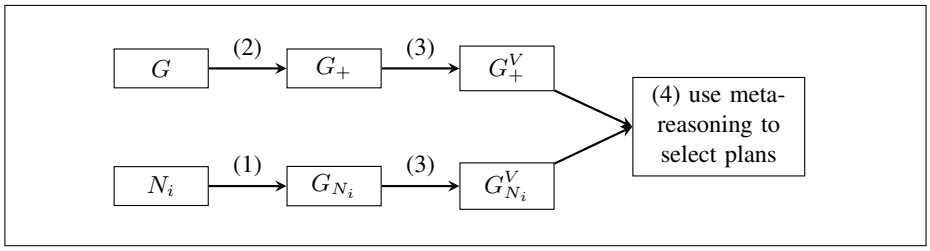

Fig. 1. Process

2. Identify possible norm violations and augment G with relevant links based on the goal, norm, and domain knowledge to obtain G_+ (§3.2).
3. Augment G_+ and G_{N_i} with valuings to obtain G^V_+ and $G^V_{N_i}$ (§3.3).
4. At run-time, use meta-reasoning to select between applicable plans (§3.4).

3.1 Translating Norms to GPTs ($N_i \rightarrow G_{N_i}$)

The first step is to convert each norm N_i into a corresponding GPT G_{N_i} by following a pattern, where the norm N_i is mapped to a GPT with a goal labelled N_i and three plans (see Figure 2).

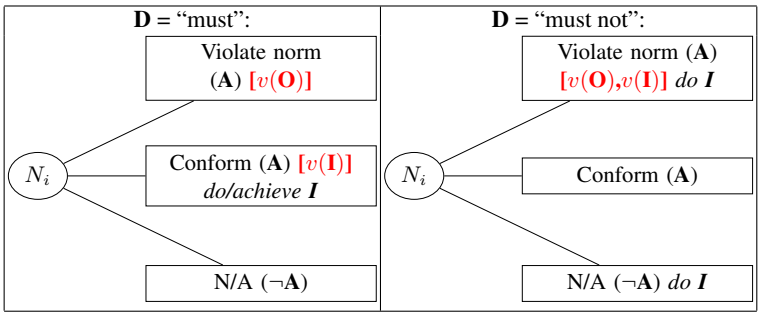

Fig. 2. Mapping Norms to GPTs. Notation: letters A,D,I,C,O correspond to those components of the norm, plan context condition is in brackets, valuings are red bold in square brackets, plan body is italic, "do/achieve" = "do" if **I** is an action, "achieve" if it is a condition, v(**I**) is the valuing of doing/achieving **I**, and v(**O**) is the valuing of **O**.

For a norm with Deontic ("**D**") "must" the first plan (intentionally) violates the norm; the second plan conforms with the norm, and may involve additional steps to do so (e.g. buying a ticket); and the third plan is applicable when the norm does not apply to the agent (e.g. in our example if it is not a paying passenger). The first two plans have the norm's attribute ("**A**") as a context condition, whereas the last plan has its negation[6].

[6] If the norm applies to all agents (when its condition holds) then the third plan can be removed.

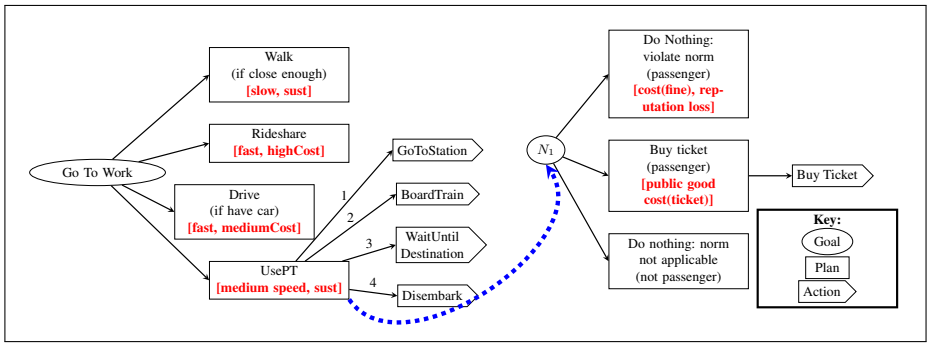

Fig. 3. GPT for the example: G (left, black only), G_+ (include blue dashed link), G_+^V (include red bold text), G_{N_1} (right, black only), $G_{N_1}^V$ (include red bold text). Notation: numbers on edges denote order, "PT" = "Public Transport", conditions are in brackets, red bold text in square brackets = valuing.

If the norm's aim ("**I**") is an action, then the second plan (to conform) includes this action. If its aim is a condition then the plan has a sub-goal to bring about the condition, and in this case additional plans need to be added (unless the agent already has plans to do so).

For a norm with deontic "must not" and an action as an aim, the pattern is similar, with the difference being that the action is associated with the first plan (violate the norm). An example would be a norm (N_2) to avoid making phone calls while using public transport. Handling "must not" norms where the aim is a condition (e.g. be in the train without ticket) requires the agent to be able to perform "look-ahead" reasoning. In order to avoid bringing about a specified condition the agent needs to be able to determine whether a particular course of action will bring about that condition *before* the course of action is selected. Unfortunately, BDI platforms do not provide this form of reasoning, making "must not" norms with a condition as aim difficult to handle (see §6). A simple solution that often works is to translate the prohibition on the condition into a prohibition on the action leading to the condition. In our example: if you have no ticket you must not board the train. This is not theoretically the same, but is a pragmatic solution that often works. For a complete theoretical discussion of this issue see [13,19].

Each plan is later (§3.3) augmented with the consequences of the choices, so that these can be used in reasoning.

For our example, the norm N_1 has a "must" deontic condition and the aim is an action. The GPT G_{N_1} is on the right of Figure 3 (ignore the red bold text for now). The context condition "passenger" comes from the norm's attribute ("**A**"), the action "Buy Ticket" in the second plan comes from the norm's aim ("**I**"), and the choice of the pattern is determined by the norm's Deontic ("**D**") being "must".

3.2 Adding links ($G \rightarrow G_+$)

The second step in the process is to identify where the agent's goals and plans (GPT G) may affect norms, and add links from G to G_{N_i}, using the norm's condition ("**C**") and domain knowledge.

For a norm N_i with deontic ("**D**") "must", the intuition is that around the point where the norm's condition ("**C**") applies, we need to trigger the GPT for the norm (G_{N_i}). For our running example the condition of the norm N_1 is "when riding public transport", so we examine the GPT G (left side of Figure 3) to find where it uses public transport.

The exact link point depends not just on the goal and norm, but also on domain knowledge: the link should be added at the point in G where the additional step(s) to conform with the norm (if any) can be performed. For our example, since conforming with N_1 requires the additional step of buying a ticket, the exact point from which G should link to G_{N_1} depends on when it is possible to buy a ticket. If tickets can be bought online at any point in time, then the link (depicted as a dashed blue line) is from the "UsePT" plan. However, if tickets can only be bought at the station, or when boarding the train, then the link needs to be from a specific point in the "UsePT" plan.

For a norm N_i with Deontic ("**D**") "must not" and aim ("**I**") that is an action, the intuition is that the norm prohibits a particular action from being done, so all occurrences of the action in G need to be replaced with a link to G_{N_i}. For example, if we have a norm N_2 prohibiting making phone calls while using public transport, then if the agent's plans G included making a phone call, then we would replace it with a link to G_{N_2} (not shown in the figure). Recall that the prohibited action (e.g. making a phone call) is in G_{N_2}, so if the agent chooses to violate the norm, it will be done by the first plan of G_{N_2}.

3.3 Adding Valuings ($G_{\ldots} \rightarrow G^{\mathbf{V}}_{\ldots}$)

The next step is to add valuings to the GPTs. These capture key attributes of the various options, including the consequences of violating norms, and of any additional steps required to conform with norms, and are used in meta-reasoning (§3.4).

We follow Cranefield *et al.* [8] in using valuings to capture an abstract evaluation of the key aspects of the effects of options. The key aspects are the ones that are relevant to decision-making, specifically to determining the desired preference over options. For example, if an agent cares about the cost of different options, then we need the valuings to include an indication of the cost associated with each option. This might be abstract (e.g. high, medium, low), rather than the actual cost, since an abstraction may suffice for reasoning.

For G_+ we add valuings based on domain knowledge. For example, that some transport options are faster. For our example we assume that the key attributes are cost and speed, as well as an indication of whether an option is considered to be environmentally sustainable, and whether a cost is considered to be contributing to the public good (for how these play a role in the reasoning see §3.4). Specifically, the valuings noted in the table in §2.3 are added (see red bold text in the left part of Figure 3).

For G_{N_i} the last plan (norm not applicable) has no valuing (since it has no effect). For Deontic "must" the plan that conforms with the norm has the valuings associated with any extra steps (e.g. the costs associated with buying a ticket). For the plan that violates the norm the valuings are based on the norm's "or else" ("**O**") part. For our example the first plan (violate the norm) has valuings based on the norm's "or else", specifically incurring a fine. The second plan has the additional cost associated with the extra step of buying a ticket. For Deontic "must not" the plan that conforms has no valuing (since it has no effect), and the plan that violates the norm has as valuings both the consequences of violating the norm, and the valuing of the action being performed.

Note that the information from the norm has been included in the resulting annotated GPTs as follows: attribute ("**A**") is the condition on the plans in G_{N_i}; deontic ("**D**") determines which pattern of plans to use for G_{N_i}; aim ("**I**") is used in one of the plans of G_{N_i}, and for a "must not" norm is also used to identify links between G and G_{N_i}; the condition ("**C**") is used to identify links; and the "or else" ("**O**") is used in the valuing for the plan in G_{N_i} that violates the norm.

3.4 Using Meta-Reasoning

Having created a goal-plan tree G_{N_i} for each norm N_i, made links between the original agent's GPT G and the norm GPTs G_{N_i} and added valuings to all of the GPTs, we are now in the position to realise norm-aware agents by using meta-reasoning.

The key idea is that the implementation of the GPTs in an agent-oriented programming language (see §4) runs as usual, except for key points where a decision of which plan to use for a given (sub-)goal is made by taking into account preferences over the valuings of the available options.

For each agent we define its *preferences* over valuings. This is simply a relation over valuings: we write $pref(v_1, v_2)$ to indicate that valuing v_1 is preferred over valuing v_2 (and we write $val(p)$ to denote the valuing associated with plan p). In order to be as flexible as possible we only constrain $pref$ to be irreflexive (i.e. $\neg\exists v : pref(v, v)$) and non-circular (i.e. $\neg\exists v : pref^*(v, v)$ where $pref^*$ is the transitive closure of $pref$). This means that we do not have to specify a strict ordering over valuings, nor we do we need to reduce preferences or valuings to numbers. We can do so if we want, for example, using a weighted sum over attributes of the valuings, but it is not required.

At run-time when a (sub-)goal is posted that has valuings associated with the different options we use the preferences to identify the best option(s). We define an option to be best if no other option is preferred over it (formally: $best(p) \equiv \neg\exists p' : pref(val(p'), val(p))$, where p and p' are options for the same (sub-)goal). We then constrain the plan selection to choose one of these best options (since the preference relation does not have to define a strict ordering over all options, there can be more than one best option). This is done using the platform's normal plan selection mechanism.

The way that we have defined G_{N_i} means that when the goal is triggered, if the agent prefers the valuing associated with conforming with the norm to the valuing associated with violating the norm, then the agent will choose to conform with the norm. On the other hand, if there is a preference for the consequences (i.e. valuing) of violating the norm, then the agent will knowingly violate the norm, having chosen to do so by considering the consequences of doing so.

Additionally, when considering the choice of options for G, the valuings are taken into account, including those associated with violating or conforming with the norm. For example, if the costs associated with driving are non-zero, then (if making decisions based entirely on cost), a non-norm-aware agent will choose to use public transport, since it is not aware of any costs associated with doing so. On the other hand, our norm-aware agent will take into account the consequences of violating the norm N_1 (possible fine) and of conforming with the norm (ticket cost), and might conclude that driving is a preferred option. In order to do so we do valuing propagation: when there is a link to a sub-goal (e.g. from UsePT to N_1 in Figure 3) then we augment the valuing of the source of the link (UsePT) with the valuings of the destination (N_1).

For our example we assume the following preferences, which are defined in terms of the valuing attributes of speed and cost (including a distinction between public good costs and other costs), and whether a given option is considered sustainable:

- we prefer faster options to slower ones, except that if sustainability is valued then we consider a sustainable option as being one step faster, so a non-sustainable *fast* and a sustainable *medium* speed option would be seen as being the same speed; and
- for the same speed we prefer the cheaper option, but when considering costs, we adjust the cost of public good expenditure in line with underlying values: if the agent values expenditures that support the public good then they discount the cost of such expenses (such as a train ticket) by a factor $a_v < 1$. On the other hand, if they do not value such expenditure, then we set $a_v \geq 1$.

Applying these preferences to the options in our GPTs we have two cases.

Case 1: sustainability is not valued: in this case we prefer the fast options (ridesharing and driving) over using public transport (medium speed) which in turn is preferred over walking (slow). Between driving and ridesharing, driving is preferred since it is cheaper. For the UsePT options ("PT" below), we prefer the option where no ticket is required ("NA"), and the preference between violating the norm or not depends on the value of a_v (indicated by $\lesseqgtr$ below). This gives the following ordering[7] (where $A > B$ abbreviates $pref(A, B)$):

drive $>$ rideshare $>$ PT(NA) $>$ PT(Conf) $\lesseqgtr$ PT(Viol) $>$ walk.

Case 2: sustainability is valued: in this case all the options apart from walking are deemed equivalent in speed, so the preference is based on cost, and public transport is preferred since it is cheaper, giving the following ordering:

PT(NA) $>$ PT(Viol) $\lesseqgtr$ PT(Conf) $>$ drive $>$ rideshare $>$ walk.

4 Implementation

This section explains how the conceptual design developed in the previous section can be mapped to code in an agent-oriented programming language (AOPL). We use Jason code, because it is a good representative BDI platform that is widely-used, but indicate how the approach can be used with other AOPLs. In other words, where we rely on a particular feature of Jason, we also indicate an alternative approach that is generic.

[7] Note that this ordering is total, i.e. for any A and B either $A < B$ or $B < A$, but in general this is not required.

```
+!goToWork : distance(near) & best(gtw,p1) ← !walk.
+!goToWork : best(gtw,p2) ← !rideShare.
+!goToWork : have(car) & best(gtw,p3) ← !drive.
+!goToWork : best(gtw,p4) ← !goto(station); !board(train); !wait(arrived);!disembark.

// Norm plans, first plan: violate norm
+!norm1 : passenger & best(upt,p5) ← true.
// Second plan: buy ticket to conform with norm
+!norm1 : passenger & best(upt,p6) ← !buy_ticket.
+!norm1 : (not passenger) & best(upt,p7) ← .print("Norm doesn't apply.").
```

Fig. 4. Jason code corresponding to the trees in Figure 3 (the "best" conditions are explained in §3.4)

Figure 4 shows the Jason code[8] for the goal (`goToWork`, i.e. the first 4 plans), and for the norm `norm1` (bottom three plans). For now ignore the additional context conditions "best". The code for the norm is a direct (and simple) mapping from its GPT.

In addition to these plans we need to: (1) realise the link between G^V_+ and $G^V_{N_i}$ (blue dashed line in Figure 3); (2) translate the valuings into corresponding beliefs; (3) define preferences over valuings; and (4) use the preferences to guide plan selection using meta-reasoning. Of these, only the preferences (step 3) are specific to a given agent (since different agents might have different preferences).

(1) Realising the link: this can be done in Jason using meta-events: we can specify that when the agent is about to go to the station, it needs to first check the norm. The following line of Jason code does this (`^!` indicates a meta-event, and the annotation "state(pending)" indicates when it should apply). The plan body suspends the goal `goToWork`, posts the goal $G^V_{N_i}$ (`norm1` in this case), and then resumes `goToWork`.

```
^!do(goto(station))[state(pending)] ←
    .suspend(goToWork) ; !norm1 ; .resume(goToWork).
```

A non-Jason alternative is to simply modify the 4th plan in Figure 4 (line 5) by adding `!norm1` before going to the station.

(2) Translating the Valuings: Valuings are modelled using beliefs of the form `v(`pl`,`val`)`, where pl is a plan label (each plan is given a unique label), and val is the valuing. This does not rely on specific features of Jason. For example, `v(p1, [speed(slow)])` indicates that for `p1` we have the valuing of speed being slow. Note that we implement "slow", "fast", and "medium" as `speed(x)` where `x` is the speed. We also replace costs (e.g. mediumCost, cost (fine)) with `cost(n)` where `n` is a number (e.g. 0,1,2,3) capturing the relative sizes of costs. Specifically we assume that highCost = 3, mediumCost = 2, that the cost of a ticket is 1, and that the anticipated cost of a fine is 1.5. Costs that are considered to be public good are indicated as `pgcost`.

[8] Notation: a Jason plan $t : c \leftarrow b$ has trigger t, context condition c and plan body b. Triggers include $+!g$ denoting the addition of a goal.

(3) Define preferences: This is done by defining `pref(`A`,`B`)` (i.e. valuing A is preferred to valuing B) as a collection of rules[9]. The definition must yield preferences that are non-circular and irreflexive. For our example the preferences indicated in §3 can be transcribed directly into Jason rules[10], where SA is the adjusted speed of A and CA is the cost of A (similarly for B).

```
pref(A,B) :– A≠B & get_adj_speed(A,SA) & get_adj_speed(B,SB) & SA>SB.
pref(A,B) :– A≠B & get_adj_speed(A,SA) & get_adj_speed(B,SB) & SA==SB &
    get_cost(A,CA) & get_cost(B,CB) & CA<CB.
```

Defining preferences without Jason's extensions can be done by renaming goals, e.g. `go-ToWork` to `goToWork2`, and adding plans that handle the trigger `+!goToWork` by first computing the preferences, and then posting the sub-goal `goToWork2`. Computing the preferences is done by assessing for each pair of plans A and B that handle the trigger `+!goToWork`, whether A's valuing (V_A) is preferred to B's (V_B), and, if so, asserting the belief `pref(`V_A`,`V_B`)`. This process would also be done for other goals (e.g. `norm1`). Note that the condition for checking whether V_A is preferred over V_B is included in the context condition of a plan, rather than using Jason rules.

(4) Adding meta-reasoning: In order to do meta-reasoning we need to first reify the program, so that the reasoning done to select plans has access to information about the agent's plans and their context conditions. This can be done by adding beliefs that indicate for each plan its label, what goal it achieves, and its context condition. These additional beliefs can be added as a straightforward pre-runtime pre-processing step. An alternative approach in Jason is to use Jason's meta-reasoning features, specifically using `.relevant_plan`.

Secondly, in order to implement meta-reasoning we add to each plan a context condition `best(`gl`,`pl`)` where gl is a goal label and pl is a plan label. We also then define:

```
best(G,L1) :– plan(G,L1,C1) & C1 & v(L1,V1) &
  not( plan(G,L2,C2) & L2≠L1 & C2 & v(L2,V2) & pref(V2,V1) ).
```

This says that the plan with label `L1` is a best option to achieve goal `G` if there is no other plan for `G` that is applicable (its context condition holds) where that plan's valuing is more preferred. Recall that there can be more than one best option, since we do not require that preferences give a full ordering over options.

In order to implement meta-reasoning without using Jason extensions, in this case the rule defining `best`, we can simply replace `best` in each plan's context condition by expanding the condition, so, for example, the first plan would become:

```
+!goToWork : distance(near) & v(p1,V1) & not( plan(gtw,P2,C2) & P2≠p1 & C2 & v
    (P2,V2) & pref(V2,V1) ) ← !do(walk).
```

To summarise, the norm-aware program extends the original program by: (1) adding links (one line of code for each link); (2) adding beliefs to indicate the valuings of

[9] Jason rules allow the definition of beliefs in a way similar to Prolog.

[10] The definition of `get_adj_speed` converts speed to a number (e.g. fast=2) and adds 1 if the option is sustainable ("`sust`") and sustainability is valued. The definition of `get_cost` adjusts public good costs by a_v - see the supplementary document [37] for full code.

different plans; (3) defining the preference over options; (4.1) adding beliefs about each plan; and (4.2) adding the one line to define `best` and, where meta-reasoning is needed, adding to each plan's context condition the `best` condition. **The scenario has been fully implemented and code is available [37].**

Our discussion so far has focused on Jason, both with and without its extensions to AgentSpeak. We now briefly consider some other AOPLs. Firstly, we note that the approach described for Jason without its extensions is usable with a range of other BDI AOPLs. However, some AOPLs provide features that make it easier. For example, in SARL's goal extension [22,23] context conditions are actually numerical ($0-1$) rather than Boolean, with the plan having the highest value being selected. This can allow us to realise meta-reasoning more simply: if we can define a mapping from valuings to numbers, then we simply define the context condition for a plan as either 0 (if the plan is not applicable) or computing the number for the plan's valuing in the current context. The agent platform JACK [35] provides a meta-level reasoning feature that allows the decision of which plan to select to be done by a meta-level plan responding to the posted `PlanChoice` event [1, §8.3]. Both these platforms considerably simplify adding meta-reasoning.

5 Evaluation

Our evaluation has two parts. Firstly (§5.1), we demonstrate, using our example, that our approach handles the three cases highlighted in the introduction: choosing to conform with a norm, requiring plan modification; knowingly violating a norm; and changing the choice of plan. Secondly (§5.2), we demonstrate how our framework operates at the right level of abstraction by showing that it is easy to accommodate changes, such as adding new norms. That this can be done easily, and with local changes to the relevant parts of the code, provides evidence that the approach is usable. It also shows that we can now experiment with all kinds of norms and combinations of norms to see the effect on the whole MAS, including assessing the system to detect and avoid unwanted behaviour.

5.1 Covering the Different Cases

First, note that without the norms, the behaviour of the agent is to (unknowingly) violate the norm when it chooses to use public transport. Adding the norms allows for a range of behaviours, and we now demonstrate how the three cases (§1) are handled.

Case 1: adding steps to conform with a norm. When choosing to use public transport, if the agent decides to conform with the norm due to the risk of a fine, then the execution of the Norm GPT (G_{N_1}) adds the additional step of buying a ticket.

Case 2: knowingly violating a norm. If sustainability is valued then using public transport is preferred, and the norm is relevant. In the case where the agent does not value public good expenditure (such as buying a ticket), and the perceived fine for not having a ticket[11] is low, it chooses to violate the norm, riding without buying a ticket.

[11] The perceived fine is roughly the actual fine combined in some way with an estimated probability of being fined. Note that humans tend not to do probabilistic calculations by multiplying probabilities by outcomes, hence "combined" rather than "multiplied".

This is done knowingly: the meta-reasoning process takes into account the norm's violation and its consequences, and nevertheless chooses to violate the norm.

Case 3: changing choice of plan. If sustainability is valued and the cost of driving is lower than both the (adjusted) cost of the public transport ticket and of the perceived fine for travelling without a ticket, then the agent will choose to drive rather than use public transport. It does this because of the consequences of the norm: without the norm, it would choose to use public transport[12], but taking into account the costs associated with conforming or violating the norm leads to driving being preferred in this situation.

The following table captures the course of action chosen for these and other situations. Unless noted otherwise, all situations assume that the belief `passenger` is true (i.e. the norm applies) and that distance is far. Otherwise, any beliefs not listed are false (notation: "sust" = "value_sustainability", "driveCheaper" = driving has been given a lower cost, "usePT(X)" = "use public transport", where X indicates whether the norm is not applicable ("NA"), is Viol(ated) or is Conf(ormed) with, and "-" indicates that a_v is not relevant).

Beliefs	a_v	Course of Action
have(car)	-	drive
¬have(car)	-	rideshare
sust, ¬passenger	-	usePT(NA)
sust	0.5	usePT(Conf) [includes buying a ticket] (**Case 1**)
sust	2	usePT(Viol) (**Case 2**)
sust, have(car), driveCheaper	1	drive (**Case 3**)

This demonstrates that the behaviour of our norm-aware agent covers the three cases. Firstly, when it decides to conform with the norm, it adds to its plan the step to buy a ticket (Case 1). Secondly, in some situations the agent violates the norm, but because it is doing so with awareness of the consequences, it only does so when the consequences are acceptable (Case 2). Thirdly, being norm-aware can affect the choice of plan (Case 3).

5.2 Flexibility

We now argue that our framework is flexible in that it is easy to add and modify norms, and that this does not require modifying the non-norm-related code. This allows to investigate the effects of adding norms to a MAS without a huge rewrite of the code. We briefly describe a number of possible modifications, and for each indicate what would need to change in the goal trees and the code. The supplementary document [37] includes the full code.

Change 1: fine-tuning norm applicability. We could make the norm more realistic by capturing that children below a certain age do not require a ticket, or (for some jurisdictions) that adults over a certain age do not require a ticket at certain hours of the day (off peak). These changes can be done by modifying the definition of paying passenger ("passenger"), for example:

[12] Because sustainability is valued, it discounts the difference in speed, and, ignoring the costs implied by the norm, public transport is cheaper than driving.

```
// not paying passenger iff age<5 or (age>=65 and not peak)
passenger :− age(N) & N>=5 & (N<65 | peak).
```

Change 2: additional requirement. One additional real requirement is that some people are entitled to cheaper concession tickets, but they need to carry and provide proof of this entitlement (e.g. a valid student card). This requirement can be modelled by adding a norm (N_3) to indicate that concession passengers must have a valid ticket and carry their concession card when riding public transport or else pay a fine.

Following our framework, this is modelled by adding a new GPT for the norm (see [37, Figure 1]) with four options: (1) violate the norm by not buying a ticket (condition: concession card holder, abbreviated to "ccholder", and valuing [cost(fine)]); (2) violate the norm by buying a concession fare while not having the concession card (condition: ccholder and not having the concession card; valuing: [cost(concFine)[13] + public good cost(concTicket), reputation loss]; (3) conform by buying a concession ticket while having a concession card (condition: concession card holder, having a concession card; valuing: [public good cost(concTicket)]); and (4) do nothing if norm not applicable (not a concession card holder). There are 4 options rather than 3 because there are two ways of violating the norm: travelling without any ticket, or travelling with a concession ticket but no concession card. Additionally, we could also add another option to conform with the norm without a concession card by buying a full-price ticket.

The additional code is in the supplementary document (§1.7, lines 14-41). We highlight that no changes are required to existing code other than adapting the link to include the new norm and modifying the definition of passenger to exclude concession travellers. The new code added corresponds directly to the GPT, in the same way that the code previously presented corresponds to the other Norm GPT: each option is a plan with the indicated condition and action, and we add beliefs to capture the indicated valuings.

Change 3: methods of buying tickets. Different public transport systems provide a range of options for buying tickets, including buying them on the train (or when boarding using a credit card to board), having to buy them at the station before boarding, or being able to buy them online at any point in time. These options can be modelled by adjusting the link to trigger the norm at the point where a ticket can be bought.

Change 4: dealing with plan failure. We now (briefly) demonstrate how our approach can handle failure. Suppose that we have the option to work from home (WFH), modelled as a parent goal "work", with two plans: working from home, and going to work (using the existing plans; see [37, Figure 2]). We assume that the option to WFH is subject to norms, such as limiting it to 3 days a week and that certain meetings are preferred to be face-to-face. This means that we prefer to go to the office under these situations (but we assume that otherwise we prefer to work at home). We also model that in the event of going to the office failing, we might be running late, and that being on time to a meeting is more important (hence overrides) the preference for being in the office. To summarise, we have two preferences: we prefer to WFH (unless we have done it too much or would miss a preferably face-to-face meeting), and, unless we are running late to a meeting, we prefer to avoid violating those norms relating to

[13] We assume that inspectors might be less likely to fine someone for not having a concession card, in which case the anticipated fine cost(concFine) would be less than cost(fine).

the WFH option. In a scenario where attempting to work results in going to work (to avoid violating a WFH-related norm), but this fails (resulting in running late), the agent then selects WFH, despite this violating the WFH-related norm. Code for this is in the supplementary document [37, §1.7, lines 58-75].

6 Discussion

We presented an approach for engineering norm-aware BDI agents that can be used with existing AOPLs without requiring them to be modified. Our approach includes a simple process for translating norms into goal-plan trees, and a method for implementing these in Jason. In addition to showing how we can implement norm-aware agents in Jason, we also discuss how this can be done in "vanilla" AgentSpeak (i.e. without using Jason's extensions), and briefly indicate how specific features of SARL and of JACK can make implementation easier when using these platforms.

Future work includes:

1. **Automate the generation of code:** Except for the preferences (step 3) which are necessarily domain-specific and possibly also agent-specific, the other code required could be generated from the GPT in a straightforward manner.
2. **Automate the identification of where links are required:** In our process the developer needs to consider the original GPT in order to identify where norms might be violated. This could be automated if we had additional information on the effects of the agent's actions. However, providing this information is a significant additional requirement, so it is not clear that the benefit of being able to identify links is worth the additional burden.
3. **Develop a larger norm-aware BDI system:** We have "closed the loop" with an implementation that shows that our approach works. Developing a more complex system would (i) demonstrate that the approach works with larger and more complex scenarios, and (ii) provide a basis for assessing the run-time overhead. As a useful example we could look at the agent-based social simulation of the COVID crisis [12]. It already has a very rudimentary way for the agents to follow or violate norms, but needs to be redesigned for each new norm.
4. **Extend our approach to handle "must not" norms with a condition as Aim:** We could build on approaches to adding look-ahead-planning to BDI agents [26] to handle these sorts of norms, and it may be possible to handle some cases by generating and propagating summary information about the effects of actions [29,33,31,32].
5. **Explore the links to explanation:** There is a whole body of work devoted to making agents explainable, including work on making BDI agents explainable [38]. The addition of norms can affect explanation, since an explanation might include a norm, e.g. "I chose to walk in order to avoid a fine for using public transport without a ticket".
6. **Add Dynamic Norms:** This requires runtime plan addition.

Acknowledgments. Michael Winikoff would like to thank the University of Ljubljana for hosting him on sabbatical while parts of this paper was written.

References

1. Agent Oriented Software: Jack intelligent agents: Agent manual (2012). https://aosgrp.com.au/wp-content/uploads/2022/08/JACK-Agent-Manual.pdf
2. Alechina, N., Dastani, M., Logan, B.: Programming norm-aware agents. In: van der Hoek, W., Padgham, L., Conitzer, V., Winikoff, M. (eds.) International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes). pp. 1057–1064. IFAAMAS (2012). http://dl.acm.org/citation.cfm?id=2343848
3. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with jacamo. Sci. Comput. Program. **78**(6), 747–761 (2013). https://doi.org/10.1016/J.SCICO.2011.10.004
4. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. Wiley Series in Agent Technology, Wiley (2007)
5. Castelfranchi, C.: Formalising the informal? Dynamic social order, bottom-up social control, and spontaneous normative relations. Journal of Applied Logic **1**(1), 47–92 (2003). https://doi.org/10.1016/S1570-8683(03)00004-1
6. Castelfranchi, C., Dignum, F., Jonker, C.M., Treur, J.: Deliberative normative agents: Principles and architecture. In: Jennings, N.R., Lespérance, Y. (eds.) Intelligent Agents VI LNAI 1757. pp. 364–378. Springer (2000)
7. Chopra, A.K., Baldoni, M., Christie, S., Singh, M.P.: Azorus: Commitments over protocols for BDI agents. In: Das, S., Nowé, A., Vorobeychik, Y. (eds.) Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025. pp. 490–499. International Foundation for Autonomous Agents and Multiagent Systems / ACM (2025)
8. Cranefield, S., Winikoff, M., Dignum, V., Dignum, F.: No pizza for you: Value-based plan selection in BDI agents. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017. pp. 178–184. ijcai.org (2017). https://doi.org/10.24963/IJCAI.2017/26
9. Crawford, S.E., Ostrom, E.: A grammar of institutions. American political science review pp. 582–600 (1995)
10. Dechesne, F., Dignum, V., Tan, Y.H.: Understanding compliance differences between legal and social norms: The case of smoking ban. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 7068 LNAI, pp. 50–64 (2012)
11. Dignum, F.: Autonomous agents with norms. Artificial Intelligence and Law **7**(1), 69–79 (Mar 1999)
12. Dignum, F. (ed.): Social Simulation for a Crisis: Results and Lessons from Simulating the COVID-19 Crisis. Springer (2021). https://doi.org/10.1007/978-3-030-76397-8
13. Dignum, F., Kuiper, R.: Combining dynamic deontic logic and temporal logic for the specification of deadlines. In: 30th Annual Hawaii International Conference on System Sciences (HICSS-30), 7-10 January 1997, Maui, Hawaii, USA. pp. 336–346. IEEE Computer Society (1997). https://doi.org/10.1109/HICSS.1997.663191
14. Hechter, M., Opp, K.D. (eds.): Social Norms. Russel Sage Foundation, New York (2001)
15. Lee, J., Padget, J.A., Logan, B., Dybalova, D., Alechina, N.: N-Jason: Run-time norm compliance in AgentSpeak(L). In: Dalpiaz, F., Dix, J., van Riemsdijk, M.B. (eds.) Engineering Multi-Agent Systems - Second International Workshop, EMAS 2014, Paris, France, May 5-6, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8758, pp. 367–387. Springer (2014). https://doi.org/10.1007/978-3-319-14484-9_19
16. Malle, B.F.: How the Mind Explains Behavior. MIT Press (2004), ISBN: 9780262134453

17. Meneguzzi, F., Rodrigues, O., Oren, N., Vasconcelos, W.W., Luck, M.: BDI reasoning with normative considerations. Eng. Appl. Artif. Intell. **43**, 127–146 (2015). https://doi.org/10.1016/J.ENGAPPAI.2015.04.011
18. Meneguzzi, F.R., Luck, M.: Norm-based behaviour modification in BDI agents. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 1. pp. 177–184. IFAAMAS (2009). https://dl.acm.org/citation.cfm?id=1558037
19. Panagiotidi, S., Vázquez-Salceda, J., Dignum, F.: Reasoning over norm compliance via planning. In: Aldewereld, H., Sichman, J.S. (eds.) Coordination, Organizations, Institutions, and Norms in Agent Systems VIII - 14th International Workshop, COIN 2012, Held Co-located with AAMAS 2012, Valencia, Spain, June 5, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7756, pp. 35–52. Springer (2012). https://doi.org/10.1007/978-3-642-37756-3_3
20. van Riemsdijk, M.B., Dennis, L.A., Fisher, M., Hindriks, K.V.: Agent reasoning for norm compliance: a semantic approach. In: Gini, M.L., Shehory, O., Ito, T., Jonker, C.M. (eds.) International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013. pp. 499–506. IFAAMAS (2013). http://dl.acm.org/citation.cfm?id=2485000
21. van Riemsdijk, M.B., Dennis, L.A., Fisher, M., Hindriks, K.V.: A semantic framework for socially adaptive agents: Towards strong norm compliance. In: Weiss, G., Yolum, P., Bordini, R.H., Elkind, E. (eds.) Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015. pp. 423–432. ACM (2015). http://dl.acm.org/citation.cfm?id=2772935
22. Rodriguez, S., Gaud, N., Galland, S.: SARL: A general-purpose agent-oriented programming language. In: The 2014 IEEE/WIC/ACM International Conference on Intelligent Agent Technology. vol. 3, pp. 103–110. IEEE Computer Society Press, Warsaw, Poland (2014). https://doi.org/10.1109/WI-IAT.2014.156
23. Rodriguez, S., Thangarajah, J., Winikoff, M.: A behaviour-driven approach for testing requirements via user and system stories in agent systems. In: Agmon, N., An, B., Ricci, A., Yeoh, W. (eds.) Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023. pp. 1182–1190. ACM (2023). https://dl.acm.org/doi/10.5555/3545946.3598761
24. dos Santos Neto, B.F., da Silva, V.T., de Lucena, C.J.P.: Using jason to develop normative agents. In: da Rocha Costa, A.C., Vicari, R.M., Tonidandel, F. (eds.) Advances in Artificial Intelligence - SBIA 2010 - 20th Brazilian Symposium on Artificial Intelligence, São Bernardo do Campo, Brazil, October 23-28, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6404, pp. 143–152. Springer (2010). https://doi.org/10.1007/978-3-642-16138-4_15
25. dos Santos Neto, B.F., da Silva, V.T., de Lucena, C.J.P.: Developing goal-oriented normative agents: The NBDI architecture. In: Filipe, J., Fred, A.L.N. (eds.) Agents and Artificial Intelligence - Third International Conference, ICAART 2011, Rome, Italy, January, 28-30, 2011. Revised Selected Papers. Communications in Computer and Information Science, vol. 271, pp. 176–191. Springer (2011). https://doi.org/10.1007/978-3-642-29966-7_12
26. Sardiña, S., Padgham, L.: A BDI agent programming language with failure handling, declarative goals, and planning. Auton. Agents Multi Agent Syst. **23**(1), 18–70 (2011). https://doi.org/10.1007/S10458-010-9130-9
27. Savarimuthu, B.T.R., Purvis, M.A., Purvis, M.K.: Social norm emergence in virtual agent societies. In: AAMAS '08: Proceedings of the 7th international joint conference on Au-

tonomous agents and multiagent systems. pp. 1521–1524. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2008)
28. Schwartz, S.: An Overview of the Schwartz Theory of Basic Values. Online Readings in Psychology and Culture **2**(1) (2012)
29. de Silva, L., Sardiña, S., Padgham, L.: Summary information for reasoning about hierarchical plans. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 1300–1308. IOS Press (2016). https://doi.org/10.3233/978-1-61499-672-9-1300
30. Thangarajah, J., Padgham, L.: Computationally effective reasoning about goal interactions. J. Autom. Reason. **47**(1), 17–56 (2011). https://doi.org/10.1007/S10817-010-9175-0
31. Thangarajah, J., Padgham, L., Winikoff, M.: Detecting & avoiding interference between goals in intelligent agents. In: Gottlob, G., Walsh, T. (eds.) IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003. pp. 721–726. Morgan Kaufmann (2003). http://ijcai.org/Proceedings/03/Papers/105.pdf
32. Thangarajah, J., Padgham, L., Winikoff, M.: Detecting & exploiting positive goal interaction in intelligent agents. In: The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003, July 14-18, 2003, Melbourne, Victoria, Australia, Proceedings. pp. 401–408. ACM (2003). https://doi.org/10.1145/860575.860640
33. Thangarajah, J., Winikoff, M., Padgham, L., Fischer, K.: Avoiding resource conflicts in intelligent agents. In: van Harmelen, F. (ed.) Proceedings of the 15th European Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002. pp. 18–22. IOS Press (2002)
34. VanHee, L., Aldewereld, H., Dignum, F.: Implementing norms? In: Hubner, J.F., Petit, J.M., Suzuki, E. (eds.) Proceedings International Joint Conference on Web Intelligence and Intelligent Agent Technology, pp. 13–16 (2011)
35. Winikoff, M.: JACK intelligent agents: An industrial strength platform. In: Multi-Agent Programming, pp. 175–193. Springer, New York, NY (2005)
36. Winikoff, M.: Implementing commitment-based interactions. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007. p. 128. IFAAMAS (2007). https://doi.org/10.1145/1329125.1329283
37. Winikoff, M., Dignum, F., Rodriguez, S., Thangarajah, J.: Supplementary material for "engineering norm-aware bdi agents" (Feb 2026). https://doi.org/10.5281/zenodo.18602838
38. Winikoff, M., Sidorenko, G., Dignum, V., Dignum, F.: Why bad coffee? explaining BDI agent behaviour with valuings. Artif. Intell. **300**, 103554 (2021). https://doi.org/10.1016/J.ARTINT.2021.103554