Anticipatory Thinking in Normative Contexts

Jomi Fred Hübner[1][0000−0001−9355−822X], Elena Yan[2][0009−0000−6660−9378], Luis G. Nardin[2][0000−0002−4506−2745], and Olivier Boissier[2][0000−0002−2956−0533]

[1] Federal University of Santa Catarina, Brazil
jomi.hubner@ufsc.br
[2] Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F-42023
{elena.yan,gnardin,olivier.boissier}@emse.fr

Abstract. This paper introduces an anticipatory norm-compliant BDI agent that considers the future consequences of its own behaviour before committing to a goal in the context of a system regulated by norms. The agent simulates the future using explicit models of the environment, itself, and the normative system, checking for norm violations if its plans are executed. When such violations are anticipated, the agent can revise its decision by selecting existing alternative plans or drop the goal if an alternative is not found. By relying solely on existing plans and avoiding dynamic plan generation, our approach preserves behavioural safety while ensuring norm compliance. An implementation demonstrates how anticipation enables agents to exploit norm activation conditions and achieve goals more efficiently while remaining compliant.

Keywords: Anticipatory Thinking · Cognitive Agents · Normative Compliance

1 Introduction

This paper motivation is illustrated by a grid-world scenario in which a single agent has the goal to arrive at a target destination. The agent has knowledge that supports its decisions about which actions will bring it from its current location to the target destination. The agent moves towards a next position that is nearest to the destination and is not occupied. While moving, the agent discovers that some places are forbidden (like a limited traffic zone – LTZ). If it simply continues following its displacement strategy, it will violate the norm. There are several ways to address this problem:

1. Reprogram the agent: The developer can foresee that situation and improve the knowledge of the agent to handle that. For instance, considering LTZ places as occupied.
2. Learn from sanctions: The agent has (reinforcement) learning capabilities. It violates the norm, receives negative rewards, and changes its knowledge to avoid the same situation in the future.

3. Norm aware agent: The agent recognises the norm and is able to foresee that its next move would violate this norm. It therefore chooses whether to proceed or not, weighing the advantages against the potential disadvantages of the expected sanction.

The first solution depends on the developers and their ability to foresee all possible scenarios. It is not suitable for open systems, where the environment is unknown at the design time. The second solution requires a try-and-error approach, which is quite costly and will be applicable just in future situations. The third solution is more suitable. However, the agent might have travelled a long path to realise it cannot continue, wasting resources.

In this paper, we generalise this problem and introduce an agent capable of *anticipating* future norm violations that would result from the execution of its current and validated strategy. Upon foreseeing a violation, the agent decides in advance an appropriate course of action. For instance, prior to initiating movement toward its destination, the agent discovers a norm, predicts that its current strategy would violate this norm, and then selects an appropriate response (e.g., replanning, aborting the goal, or deliberately move on accepting the violation) before committing resources toward achieving that destination.

Our proposal, as presented in Sec. 2, is strongly inspired by the *anticipatory thinking* proposals and our previous work on the topic [15,13]. Simulation, presented in Sec. 3, is the main technique used to foresee the future, based on given models of the agent, environment and norms. If a violation is foreseen, we propose that the agent searches for alternative options among its plans (Sec. 4). The proposal is implemented and illustrated using the grid-world scenario (Sec. 5). We discuss the proposal, its features, limitations, and related work in Sec. 6. The preliminary conclusions and future work are presented in Sec. 7.

2 An anticipatory normative agent

A definition of anticipation is proposed by Rosen [28]: "An anticipatory system is a system containing a predictive model of itself and/or its environment, which allows it to change state at an instant in accord with the model's predictions pertaining to a later instant." In the context of multiagent systems, it is important to differentiate anticipation from planning. Whereas classical planning addresses how to generate future events from the current state [12], anticipation focuses on how future states influence decisions made in the present [38,1,26]. A simple example of Anticipation Thinking (AT) is *planning* to go for a walk in the afternoon. Before leaving the house in the morning, we notice that it might rain later in the day, so we decide to bring an umbrella [2].

Based on the proposal of Jones and Laird [16], the AT process of our agent has the following steps:

1. Initialising AT: Determine whether and when thinking about the future is worthwhile. AT is an expensive process, and we should not perform it continuously. In our proposal, this step is executed together with the decision to commit to a new goal in a typical Belief Desire Intention (BDI) agent [27,34].

2. Imagining the future: Based on the current information and strategy, the agent thinks about future events and their impact on itself and the environment. The proposal for this step is to use simulation to foresee future events (Sec. 3). The impacts in which we are interested are norm violations.[3]
3. Prospective planning: Once an undesired future event is imagined, the agent revises its decisions and strategy as needed in response to future events. The proposal is to search for alternative plans in the agent plan library (Sec. 4).
4. Preparatory action: Given the prospective plan (if any), the agent decides what to do now to prepare for the future. We consider two possibilities for this step: select an alternative plan from the plan library or give up the goal (Sec. 4).

Briefly, we delimit the scope of this paper to the anticipation of future norm violation for a BDI agent that already has a set of plans. These plan if executed to achieve some goal, may violate a norm in the future. We also opted not to change or add new plans to solve the future problem. The solution should be found within the current set of plans or the goal must be dropped. The main reason for that is that we are assuming the plan library as defining a safety boundary for the agent's behaviour, and no new dynamic and unexpected behaviour is desired. All these delimitations are further discussed in Sec. 6.

The simulation of the future in our AT agent (step 2 of the process) requires that the agent knows the models of the environment, itself, and norms. These models are detailed in the next subsections. In our previous work on anticipatory thinking [15], the future problem is determined based only on the state of the environment. However, the detection of a norm violation cannot be reduced to this case: the current state of the environment is not enough to infer the current normative state. The detection of a norm violation requires both the environment and normative state. For example, consider a norm that is activated due to an environment state and later violated due to another environment state. Only this latter environment state is not enough to identify a violation case, we need to know that the norm is active. Thus, we also need a model of the normative system.

2.1 The environment model

We model the evolution of the environment as proposed in Markov Decision Processes [10]: the environment changes as a result of actions performed by an agent. Considering a set of possible environment states S, and possible actions A, the evolution of the environment is modelled by the function:

$$e : S \times A \to S$$

[3] Other cases of future problems are investigated in our previous work. In [15] the future problem is essentially an undesired state and the failure to satisfy some goal. In [13] the future problem is caused by other agents.

Given a state $s \in S$ and the action $a \in A$, $e(s, a)$ is the next state of the environment after the execution of the action. We consider the environment as static, fully observable, deterministic, and discrete [29].[4]

2.2 The agent model

The agent model is simplified to a function representing possible actions for each environment state (agent *policy* function):

$$\pi : S \rightarrow A^r$$

Given a state s, $\pi(s)$ is a sequence of r possible action *options* for the agent, ordered by *preference*. For example, in the grid-world scenario, if the state is s and $\pi(s) = \langle s, w, e, n \rangle$, the agent prefers to go south (s), then west (w), then east (e), then north (n).[5]

The agent policy function represents the knowledge the developer gave to the agent at design time: the preferred actions for every state ordered according to a strategy. This strategy does not consider norms and thus the π function does not consider the normative state. In our proposal, norms are discovered and properly handled at runtime by the AT agent.

2.3 The normative model

While the environment evolves through agent's actions, the normative state of a system evolves based on the state of the environment (the brute facts [32,7]). For the purpose of defining the AT agent, we introduce here a very simplified view of the normative dynamics, ignoring several aspects of this realm [4,35,21,6,33]. Thus, considering a set of possible normative states P, the normative state evolution is modelled by the function:

$$\rho : P \times S \rightarrow P$$

Given a normative state $p \in P$ and the environment state $s \in S$, $\rho(p, s)$ is the next normative state from the brute facts represented by s.

Two other functions are defined to assign properties to normative states. Considering the set of boolean values $\mathbb{B} = \{true, false\}$, the function v determines whether a norm is violated in a normative state and the function α determines whether a norm is activated in a normative state.

$$v : P \rightarrow \mathbb{B}$$

$$\alpha : P \rightarrow \mathbb{B}$$

[4] Here, we are considering only one agent, see [13] for a multi-agent case. We are also considering a deterministic environment, for a more uncertain environment, see [15].

[5] We use $[i]$ to denote the i-th element of a sequence and $[-1]$ to refer to its last element. For example, if $\pi(s) = \langle s, w, e, n \rangle$, then $\pi(s)_{[1]} = s$ and $\pi(s)_{[-1]} = n$.

In our proposed simplification of the normative model, we focus only on the evolution of normative states and whether these states enclose a violation or activation of norm. A normative state is just an identifier. We therefore abstract away from the internal structure of what constitutes a state (as we do for the environment model) as well as how the normative system is implemented (this system is simply abstracted by the ρ function). The advantage of this approach is that we do not rely on a specific definition of norms or normative system implementation to describe the core ideas of our proposal. Naturally, for a concrete implementation (such as the one presented in Sec. 5), all these details should be provided.

3 Simulation of the future

A simulation produces a history. We represent a history h of length m ($h \in (S \times P)^m$, $m \geq 0$) as a sequence of m states $\langle (s_1, p_1), \ldots, (s_m, p_m) \rangle$. Each history state is a pair with the environment state and its corresponding normative state. We denote the last state of history h as $h_{[-1]}$, the first term of a pair with [1] ($(s,p)_{[1]} = s$), the second term with [2] ($(s,p)_{[2]} = p$), an empty history as $\langle\rangle$, and the set of all histories as H ($H = \bigcup_{i=0}^{\infty}(S \times P)^i$). The concatenation of two histories is produced by the operator $\oplus$.

A possible history is illustrated in Fig. 1. The history $\langle (s_1, p_1), (s_2, p_2), (s_3, p_3) \rangle$ is produced by the agent following its preference departing from state (s_1, p_1). In this initial environmental state s_1, the options for the agent are $\pi(s_1) = \langle a_3, a_2, a_1 \rangle$ and the preference is for action a_3 ($\pi(s_1)_{[1]} = a_3$). The next environmental state is $e(s_1, a_3) = s_2$. The corresponding normative state is given by $\rho(p_1, s_2) = p_2$. In the last history state (s_3, p_3), we can notice that a norm is violated ($v(p_3) = true$) and we thus obtain an undesired history.

Given the three models (agent, environment, norms), an agent can use the following $matrix : S \times S \times P \to H$ function to foresee whether a goal g ($g \in S$) will be achieved without any norm violation from the current state s:

$$
\begin{aligned}
matrix(g, s, p) &\stackrel{\text{def}}{=} mah(g, \langle (s, p) \rangle) \\
mah(g, h) &\stackrel{\text{def}}{=} \begin{cases} \langle\rangle & \text{if } v(p') \\ h & \text{if } s' = g \\ mah(g, h \oplus \langle (s'', p'') \rangle) & \text{otherwise} \end{cases} \\
& h' = h_{[-1]} \qquad s' = h'_{[1]} \qquad p' = h'_{[2]} \\
& s'' = e(s', \pi(s')_{[1]}) \\
& p'' = \rho(p', s'')
\end{aligned}
$$

The *matrix* function begins with an agent goal g, an environment state s and a normative state p and builds the history incrementally by simulating the future step by step starting from the last state of a history h. It considers that the agent will select its first preferred option ($\pi(s')_{[1]}$). The function returns either a successful history or an empty history if a violation is detected. The auxiliary

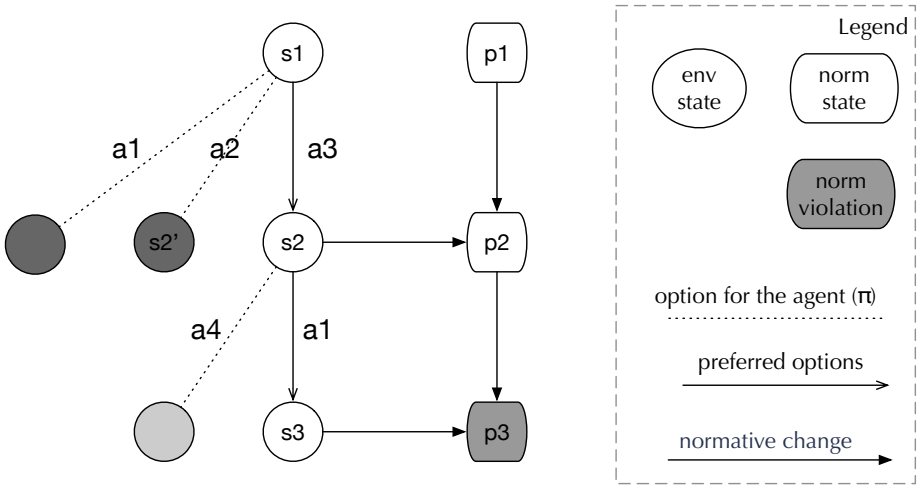

Fig. 1. Example of future simulation. The current environmental state is s_1 and the normative state is p_1. Four actions are defined ($A = \{a_1, a_2, a_3, a_4\}$), but only three are options for the agent in state s_1 ($\pi(s_1) = \langle a_3, a_2, a_1 \rangle$) and two for state s_2 ($\pi(s_2) = \langle a_1, a_4 \rangle$). The state s_1 is transformed into state s_2 by action a_3 ($e(s_1, a_3) = s_2$). The normative state p_1 is transformed into state p_2 by the new environmental state s_2 ($\rho(p_1, s_2) = p_2$).

function $mah : S \times H \rightarrow H$ maps a goal g and a history of decisions given by π, e, and ρ into a history where g is the last state.[6]

Given these definitions, we have a discrete notion of time that evolves based on changes in states produced by the actions of the agent. By simulating the future with functions e, π, and ρ, the future is linear, as well as the computational complexity to build it. We are therefore able to compute *the* future instead of the *possible* futures. This is possible because we simulate the future based only on the first option of the agents' policy. In other techniques, such as automated planning [12], the future is branching for each possible action because the agent policy is not considered (indeed, their objective is to create a policy), which usually leads to exponential complexity.[7]

4 Reasoning About Alternative Options

For the prospective planning step of the AT process, our proposal is to verify if an alternative option to achieve the agent goal will avoid the violation. Considering Fig. 1, we can verify whether actions a_2 or a_3, the second and third options

[6] If both conditions of *mah* hold (i.e., $v(p') = true$ and $s' = g$), the function returns $\langle\rangle$. The violation precedes the achievement.

[7] The computational analysis of the proposal is further elaborated in [15].

of the agent, would bring a preferable future. More specifically, the agent may select an alternative option for the current state s (given by $\pi(s)_{[j]}$ with $j > 1$) instead of the most preferred option (given by $\pi(s)_{[1]}$). Note that alternatives should be evaluated for next states as well, for instance, the options for s_2. The approach is to exploit the current agent plans, instead of creating new ones.

When our agent has a new goal g (defined as a particular environment state), it can use Alg. 1 to decide whether to commit to it or not: given the current states s and p, use the *matrix* function to foresee if the goal g is achieved in the future ($matrix(g, s, p) \neq \langle\rangle$). If so, commit to it; otherwise, search for alternative options. If no option exists, the agent drops the goal since it is preferable to do nothing rather than pursuing a goal that will violate a norm.[8] If the agent has options, select the best. The search algorithm on line 5 returns a set of alternative plans pls to achieve g without violation. The search is based on Breadth-First Search (BFS) where the nodes of the tree search are expanded by the options given by π (more details can be found in [15]). A plan is a sequence of choices of options. For instance, the plan $\langle 2, 1, ...\rangle$ means: select the second option for the first state ($\pi(s_1)_{[2]}$), the first option for the second state ($\pi(s_2)_{[1]}$), ...[9] Several criteria could be used to define the *best* function on line 8. Currently, we propose to select plans from pls that deviate less from the agent preference, i.e., that minimizes the sum of the numbers in the plan.

```
   Input: goal g, plan library pl, environment model e, normative model ρ
 1 π ← a policy for g based on pl ;
 2 matrix ← a matrix simulator based on e, ρ and π;
 3 (s, p) ← perception() ;      // the agent perceives both environment and
    normative states
 4 if matrix(g, s, p) = ⟨⟩ then
 5 |   pls ← search(g, s, p, π) ;        // pls is a set of alternative plans
 6 |   if pls = ∅ then
 7 |   |   return drop g
 8 |   b ← best(pls) ;
 9 |   π ← a new policy based on plan b ;
10 return π
```

Algorithm 1: Search for Alternatives.

The alternative plans (set pls in the algorithm) can achieve the goal without violation due to not producing states that either (i) violate any norm or (ii) (de)activate a norm. We are particularly interested in investigating when one case is better than another. For instance, if we are at 4:45pm and the LTZ is

[8] Note that we are not considering the case where the agent decides to violate, that case will be addressed in future works.

[9] We call these plans alternative plans because they are not new plans, but just a different sequence of choices from options given by the existing plans.

forbidden from 5pm until 7pm, maybe the best option is to wait a bit and use the faster path through the LTZ. This decision affects the selection of the best alternative in line 8 and is evaluated in the next section.

5 Grid World Scenario

We further discuss our proposal using the example introduced in Sec. 1. The scenario is illustrated in Fig. 2. The white path, crossing the LTZ, represents the path the agent uses if norms are ignored. This is also the shortest path.

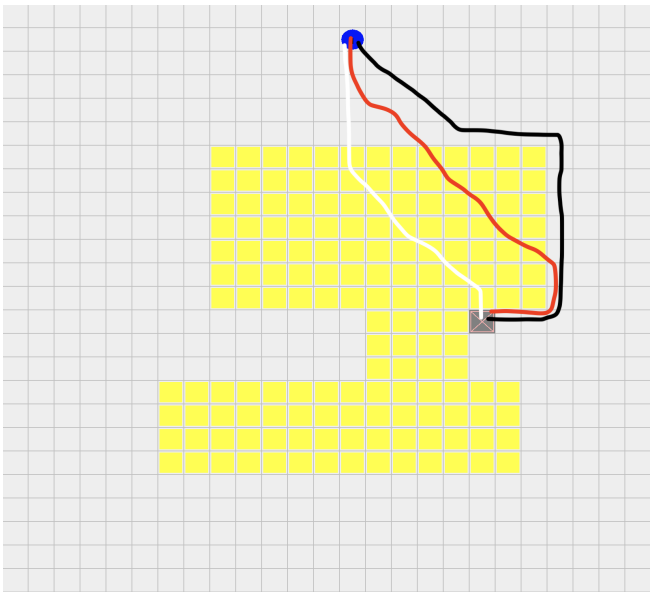

Fig. 2. The LTZ Scenario. The blue circle is the agent initial location and the grey square its destination. Yellow squares represent the LTZ.

The experiment is implemented in Jason [5][10] and part of the source code is shown in Fig. 3. The plans are there just to move the agent towards its target. There is no reference to norms, since they are discovered at runtime.

To include norms in the experiment, we use the NPL language [14,39][11], since it offers an implementation already integrated with Jason. We thus add the following norm to the system:

[10] The source code is available at https://github.com/jomifred/future-bdi. To implement the *matrix* function, the mental state of the agent is cloned and placed to run in a simulated environment instead of the real environment. The normative system is also cloned and its dynamics is based on the simulated environment.

[11] https://github.com/moise-lang/npl

```
// plans for the goal !pos(X,Y)
@  [preference(0),cost(0.0)] +!pos(X,Y) : pos(X,Y).
@s [preference(D),cost(1.0)] +!pos(X,Y) : ok(s) & distance(s ,D) <- s;  !pos(X,Y).
@sw[preference(D),cost(1.4)] +!pos(X,Y) : ok(sw)& distance(sw,D) <- sw; !pos(X,Y).
@se[preference(D),cost(1.4)] +!pos(X,Y) : ok(se)& distance(se,D) <- se; !pos(X,Y).
@w [preference(D),cost(1.0)] +!pos(X,Y) : ok(w) & distance(w ,D) <- w;  !pos(X,Y).
@e [preference(D),cost(1.0)] +!pos(X,Y) : ok(e) & distance(e ,D) <- e;  !pos(X,Y).
@n [preference(D),cost(1.0)] +!pos(X,Y) : ok(n) & distance(n ,D) <- n;  !pos(X,Y).
@nw[preference(D),cost(1.4)] +!pos(X,Y) : ok(nw)& distance(nw,D) <- nw; !pos(X,Y).
@ne[preference(D),cost(1.4)] +!pos(X,Y) : ok(ne)& distance(ne,D) <- ne; !pos(X,Y).
@id[preference(D),cost(0.9)] +!pos(X,Y) :     distance(idle,D) <- idle; !pos(X,Y).

// checks if going to some direction is possible (free cell)
ok(D) :- next(D,X,Y) & free(X,Y).

next(s ,X  ,Y+1) :- pos(X,Y). // my next location if doing south
next(sw,X-1,Y+1) :- pos(X,Y).
next(se,X+1,Y+1) :- pos(X,Y).
next(w ,X-1,Y  ) :- pos(X,Y).
next(e ,X+1,Y  ) :- pos(X,Y).
next(n ,X  ,Y-1) :- pos(X,Y).
next(nw,X-1,Y-1) :- pos(X,Y).
next(ne,X+1,Y-1) :- pos(X,Y).
next(idle,X,Y)   :- pos(X,Y).

free(X,Y) :- X >= 0 & Y >= 0 & w_size(W,H) & X < W & Y < H
             & not obstacle(X,Y)
             & not agent(_,X,Y).
distance(Dir,Dist) :- next(Dir,X,Y) & destination(GX,GY) &
                      Dist = math.sqrt( (X-GX)**2 + (Y-GY)**2 ).
```

Fig. 3. Jason code of the agent.

```
norm n1
   :  step(S) & S > 10 & S < 1000
   -> prohibition(A,n1, pos(A,X,Y) & ltz(X,Y), false).
```

the activation condition (after `:`) states that the norm is active only after step 10 and before step 1000 (simulating the time interval of the prohibition). While it is active, the norm specifies that every agent `A` is forbidden to be in a location `X`, `Y` (`pos(A,X,Y)`) that lies within the LTZ. Considering that, in the figure, the black line represents the path that never places the agent inside the LTZ. The red line represents the path that uses the LTZ up to step 10, resulting in a shorter path than the one shown in black.

In this environment, given the agent policy derived from the plans and the NPL norm that defines the normative model, our proposed agent initially detects (line 4 of Alg. 1) that following its preferred options would result in a norm violation (the white path). It then searches for alternative options (line 5). In this experiment, the best alternative (line 8) keeps the agent strategy (its preferred options) and does not violate the norm, either by not entering the LTZ (the case in the end part of the red path) or by not activating the norm (the case in the beginning part of the red path). This alternative yields a policy (line 9) that moves the agent along the red path shown in Fig. 2 (which is faster than the black path that avoids the LTZ).

This example illustrates that, in certain scenarios, it is preferable to exploit the activation condition of norms in order to find a better alternative to be norm compliant.

From an agent development perspective, we have an agent proposal that, at runtime, faces some norms and adapts its behaviour accordingly. It is neither originally programmed considering norms nor reprogrammed when (new) norms are discovered. For instance, by merely changing the norm activation condition to:

```
norm n1
   :  step(S) & S < 10
   -> prohibition(A,n1, pos(A,X,Y) & ltz(X,Y), false).
```

i.e., the LTZ is active just in the beginning of the experiment, the agent will adapt its behaviour exploiting the new norm and the idle action, resulting in a path like shown in white in Fig. 2 (it stays on idle for 6 steps and then starts moving). Furthermore, by simply adding norms into a system, newly arriving agents of the type proposed here can be constrained so that they adhere to the system's rules.

Obligations can also be used in the norms. For instance, suppose that a kind of portal has to be visited if there is one near the destination. This case can be expressed by the following norm in NPL language:

```
norm n2
    : destination(DX,DY) & portal_near(DX,DY, PX,PY)
   -> obligation(A, n2, visited(A,PX,PY), pos(A,DX,DY) ).

portal_near(DX,DY, PX,PY) :-
      portal(PX,PY) &
      D = math.sqrt((DX-PX)**2 + (DY-PY)**2) &
      D < 5.
```

The activation condition is based on the agent destination goal (`destination(DX, DY)`) and a portal near that destination. The predicate `portal_near(DX,DY, PX,PY)` is true if there is a portal at `PX, PY` and it is near the destination, as defined by the rule that follows the norm. If there is such a portal, the agent is obliged to visit its location `PX, PY` (the third argument of the obligation) before arriving at its destination. Arriving at the agent destination (`pos(A,DX,DY)`) is the obligation deadline (its fourth argument). If the deadline condition holds before fulfilling the obligation (`visited(A,PX,PY)`), the norm is considered violated in NPL.

When simulating the future with this norm, violations are foreseen as before. If the agent destination goal is near a portal (based on the environment state represented by `destination(DX, DY)` and `portal(PX, PY)`), the norm is activated in the corresponding normative state. An environment state that holds `pos(A,DX,DY)` and not `visited(A,PX,PY)` count-as a normative state with a

violation (based on function v) and that state should be avoided by our AT agent.

The particularities of NPL and its possible types of norms are irrelevant for our proposal. This language is used as an experimental tool to allow us to implement the proposal. Our proposal relies on any normative engine that computes violation of norms based on the environmental state.

In this paper, we are looking for improvements in agent decision-making with respect to potential violation of norms in the *future*. A fundamental feature of the proposal for addressing this issue is that it relies on the knowledge provided by the developer to the agent. This knowledge equips an agent with options and strategies to achieve their goals, as represented by its π function. We therefore assume that the developer is a domain expert and the agent benefits from having more options given to it. The search for alternatives algorithm exploits this practical knowledge to better choose between options and prevent problems in the future. As mentioned earlier, another motivation for this approach is to prevent the agent from creating new plans, which could imply a possible undesired behaviour.

6 Discussion and Related Work

In terms of related work, we begin by discussing the similarities between our proposal and automated planning [12], including approaches that combine planning with BDI, such as [31,22]. Planning can address norms by adding them in the action and state specifications [25,18]. While both planning and our proposal use a model of the environment to simulate the future, ours additionally considers a model of the agent itself. This agent model enables us to foresee potential future violations within a linear computational complexity (cf. *matrix* function)[12], in contrast to the typically exponential complexity associated with planning. Moreover, we assume that plans are carefully defined and verified at design time, avoiding the potential undesired surprises of dynamic generation of plans.

In the context of normative reasoning, several works tackle the problem of norm compliance from an agent perspective. Some of them focus on deciding whether or not to follow a norm [8,3,20,9], which, although complementary, is not the main problem addressed here. We focus on avoiding norm violations with anticipation.

We propose to classify related work considering how far the future is anticipated. In the context of cognitive agents, three classes are observed:

[12] The complexity of the *matrix* function is linear, since it simulates *the* future based on the first preference of the agent. However, the complexity of the search for alternative options (line 5 of Alg. 1) is exponential.

1. The action class: the agent foresees that the action it chose to execute will violate some norm (scenario 3 of the introduction). Most works can be placed in this class (e.g., BOID [8], NoA [17][13], [19]).
2. The plan class: before selecting a plan, the agent evaluates if that plan will violate some norm and decides for that plan accordingly (e.g., Nu-BDI [23], NBDI [30]). This solution depends on the granularity of plans: a long plan implies more anticipation.
3. The goal class: before committing to a goal, the agent evaluates *all* plans that will be selected to foresee if a norm will be violated. We did not find other work on this class.[14]

The plan class where goals can be achieved by only one plan can be merged with the goal class. However, we are more interested in agents where plans are divided in sub-plans, so that the agent is able to select the most suitable sub-plan for the current circumstances as it advances in the goal achievement (as is the case of several agent programming languages inspired by the Procedural Reasoning System (PRS) proposal [11]).

This classification raises the question of how far ahead the agent should anticipate. The action class is clearly easy to implement and requires little computational effort. However, the agent may spend considerable energy only to realize, too late, that the goal is unachievable (as in scenario 1 described in the Introduction). At the other extreme, the goal class can detect in advance that a goal will violate a norm, but it is more complex in terms of required models and computational cost. The plan class occupies an intermediate position between these two. Another aspect to take into account is that our simulation of the future assumes a perfect model of the environment and assumes that only our agent changes it. If these assumptions are relaxed, our confidence in the predicted future decreases the further we look ahead (as analysed in [15]). Thus, the decision of how far to anticipate should be balanced by the certainty of future states. As is often the case, the answer to the question depends on the characteristics of the application. The contribution of this paper is to offer developers a set of concepts and factors to take into account.

Regarding the proposals on anticipatory thinking that inspired our research, a contribution of this paper relies on combining agents, environment and norms in a practical implementation that allows us to experimentally investigate the

[13] Although the authors propose a decision based on the selection of plans, their definition of plan is the same as our definition of action, so NoA is placed in the action class.

[14] NBDI addresses goals, but not all plans are used to achieve them. It compares the state that the norm forbids against the agent's goals to identify a conflict. For example, if the norm prohibition is `pos(10,20) & ltz(10,20)` and the agent goal is `pos(40,50)`, there is no conflict comparing these terms and both the norm and the goal are accepted. However, the case that the agent plans to move the agent through the LTZ is not captured by NBDI when evaluating goals, although it will be when evaluating plans.

approach. As in [1], we introduce a metacognitive mechanism, and, following [16], this mechanism operates in a way that is similar to the usual agent reasoning. Our agent places a clone of itself, the environment and the normative system in a "matrix" simulator, "watches" the outcomes and changes its plans accordingly (metacognition). The reasoning of the cloned agent is identical to main agent (despite that it does not foresee the future).

7 Conclusions and Future Work

This paper introduces the initial results of our investigation of integrating anticipatory thinking with normative reasoning for BDI agents. The main contribution is an agent proposal that is capable of foreseeing, before committing to a goal, whether some norm will be violated. If that is the case, the agent is also capable of avoiding either the activation or the violation of the norm by selecting other plans available to the agent. The developer of this agent does not need to consider norms while coding the plans. The norms are discovered at runtime and used by the agent reasoning mechanism to improve its decisions.

The proposal, however, has some limitations that will be addressed in future works.

1. The requirement of the environment and normative models. This requirement is quite common in proposals that consider the future (e.g., Nu-BDI defines that model in terms of the pre-conditions and effects of actions). Besides being sometimes unrealistic, this model can be difficult to define [36]. One alternative is to learn an environment model [37]. The normative model, as proposed here, is usually the NPL program the system already has. However, the normative model depends on quality of the environment model (the brute facts) to provide good predictions. We also plan to relax some current assumptions of the environment model: deterministic, static, and fully observable.
2. Norm adoption. The focus on the anticipation process, the proposed agent does not violate a norm, preferring to change its plans or drop its goals instead. We plan to extend the agent to deliberate about violating a norm in case it conflicts with its goals or requires a lot of resources to be fulfilled. The work of n-BDI [9] is a good starting point.
3. Replanning. To prevent any violations, we suggest searching among the alternative plans that the agent already possesses. However, the agent might not have alternatives or they are costly. In these cases, adding new plans or changing current plans could be an alternative to investigate.
4. More scenarios. Although the grid scenario allowed us to perform initial experiments, it is simple compared to real applications. We plan to apply anticipation techniques in robotic applications of our laboratory [24].

Acknowledgments. The first author acknowledges the financial support from the National Council for Scientific and Technological Development (CNPq, Brazil), Grant No. 305701/2025-8.

References

1. Amos-Binks, A., Dannenhauer, D.: Anticipatory thinking: A metacognitive capability. In: Proc. of the Workshop on Cognitive Systems for Anticipatory Thinking (2019). https://doi.org/10.48550/arXiv.1906.12249
2. Amos-Binks, A., Dannenhauer, D., Gilpin, L.H.: The anticipatory paradigm. AI Magazine **44**(2), 133–143 (2023). https://doi.org/10.1002/aaai.12098
3. Andrighetto, G., Villatoro, D., Conte, R.: Norm internalization in artificial societies. AI Commun. **23**, 325–339 (2010)
4. Balke, T., da Costa Pereira, C., Dignum, F., Lorini, E., Rotolo, A., Vasconcelos, W., Villata, S.: Norms in MAS: Definitions and Related Concepts. In: Andrighetto, G., Governatori, G., Noriega, P., van der Torre, L.W.N. (eds.) Normative Multi-Agent Systems, Dagstuhl Follow-Ups, vol. 4, pp. 1–31. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2013). https://doi.org/http://dx.doi.org/10.4230/DFU.Vol4.12111.1, http://drops.dagstuhl.de/opus/volltexte/2013/3998
5. Bordini, R.H., Hübner, J.F., Wooldrige, M.: Programming Multi-Agent Systems in AgentSpeak using *Jason*. Wiley Series in Agent Technology, John Wiley & Sons, England (2007). https://doi.org/10.1002/9780470061848, https://jason-lang.github.io
6. de Brito, M., Hübner, J.F., Bordini, R.H.: Programming institutional facts in multi-agent systems. 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. LNCS, vol. 7756, pp. 158–173. Springer (2013). https://doi.org/10.1007/978-3-642-37756-3_10
7. de Brito, M., Hübner, J.F., Boissier, O.: Situated artificial institutions: stability, consistency, and flexibility in the regulation of agent societies. Autonomous Agents and Multi-Agent Systems **32**(2), 219–251 (2018). https://doi.org/10.1007/s10458-017-9379-3, https://doi.org/10.1007/s10458-017-9379-3
8. Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., van der Torre, L.: The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In: Müller, J.P., Andre, E., Sen, S., Frasson, C. (eds.) Proceedings of the Fifth International Conference on Autonomous Agents. pp. 9–16. ACM Press, Montreal, Canada (2001), citeseer.ist.psu.edu/broersen01boid.html
9. Criado, N., Argente, E., Noriega, P., Botti, V.: Reasoning about norms under uncertainty in dynamic environments. International Journal of Approximate Reasoning **55**(9), 2049–2070 (2014). https://doi.org/10.1016/j.ijar.2014.02.004
10. Garcia, F., Rachelson, E.: Markov Decision Processes, chap. 1, pp. 1–38. John Wiley & Sons, Ltd, UK (2013). https://doi.org/https://doi.org/10.1002/9781118557426.ch1
11. Georgeff, M.P., Lansky, A.L.: Reactive reasoning and planning. In: Proc. of the Sixth National Conference on Artificial Intelligence (AAAI 87). pp. 677–682 (1987)
12. Ghallab, M., Nau, D., Traverso, P.: Automated Planning and Acting. Cambridge University Press, Cambridge (2016)
13. Hübner, J., Burattini, S., Ricci, A., Mayer, S.: Anticipatory thinking in multi-agent contexts. In: Anais do XIX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações. pp. 1–12. SBC, Porto Alegre, RS, Brasil (2025). https://doi.org/10.5753/wesaac.2025.37525, (best paper)
14. Hübner, J.F., Boissier, O., Bordini, R.H.: A normative programming language for multi-agent organisations. Annals of Mathematics and Artificial Intelligence **62**(1-2), 27–53 (2011). https://doi.org/10.1007/s10472-011-9251-0

15. Hübner, J.F., Burattini, S., Ricci, A., Mayer, S.: Reflexive anticipatory reasoning by BDI agents. Auton. Agents Multi Agent Syst. **39**(1), 7 (2025). https://doi.org/10.1007/S10458-025-09687-8, https://tinyurl.com/t7jst2rm
16. Jones, S.J., Laird, J.E.: A cognitive architecture theory of anticipatory thinking. AI Magazine **44**(2), 155–164 (Jun 2023). https://doi.org/10.1002/aaai.12102, http://dx.doi.org/10.1002/aaai.12102
17. Kollingbaum, M.J., Norman, T.J.: Informed deliberation during norm-governed practical reasoning. In: Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.) Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems. pp. 183–197. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
18. Krzisch, G., Meneguzzi, F.: Planning in a normative system. In: Proc. of International Workshop on Coordination, Organisations, Institutions and Norms (COIN@AAMAS) (2017), https://www.meneguzzi.eu/felipe/pubs/coin-norm-planner-2017.pdf
19. Lee, J., Padget, J., Logan, B., Dybalova, D., Alechina, N.: Run-time norm compliance in bdi agents. pp. 1581–1582 (05 2014). https://doi.org/10.65109/YKYG6804
20. López y López, F., Luck, M., d'Inverno, M.: A normative framework for agent-based systems. Computational & Mathematical Organization Theory **12**, 227–250 (2006)
21. Mahmoud, M.A., Ahmad, M.S., Mohd Yusoff, M.Z., Mustapha, A.: A review of norms and normative multiagent systems. The Scientific World Journal **2014**(1), 684587 (2014). https://doi.org/https://doi.org/10.1155/2014/684587, https://onlinelibrary.wiley.com/doi/abs/10.1155/2014/684587
22. Meneguzzi, F., De Silva, L.: Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning. The Knowledge Engineering Review **30**(1), 1–44 (2015). https://doi.org/10.1017/S0269888913000337
23. Meneguzzi, F., Vasconcelos, W., Oren, N., Luck, M.: Nu-bdi: norm-aware bdi agents. In: Proceedings of the 10th European Workshop on Multi-Agent Systems, Dublin, Ireland (2012)
24. de Oliveira Silvestre, I., Becker, L.B., Fisher, M., Hübner, J.F., de Brito, M.: Enhanced agent-oriented programming for robot teams. Engineering Applications of Artificial Intelligence **158**, 111390 (2025). https://doi.org/10.1016/j.engappai.2025.111390, https://www.sciencedirect.com/science/article/pii/S0952197625013922
25. 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. pp. 35–52. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37756-3_3
26. Poli, R.: An introduction to the ontology of anticipation. Futures **42**(7), 769–776 (2010). https://doi.org/https://doi.org/10.1016/j.futures.2010.04.028, https://www.sciencedirect.com/science/article/pii/S0016328710000753, special Issue: Landscape Visions
27. Rao, A.S., Georgeff, M.P.: BDI agents: from theory to practice. In: Lesser, V. (ed.) Proceedings of the First International Conference on MultiAgent Systems (ICMAS'95). pp. 312–319. AAAI Pess, San Francisco, USA (1995)
28. Rosen, R.: Anticipatory Systems. Pergamon (1985). https://doi.org/https://doi.org/10.1016/C2009-0-07769-1
29. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, New Jersey, 3 edn. (2010)

30. dos Santos Neto, B.F., da Silva, V.T., de Lucena, C.J.: Developing goal-oriented normative agents: The NBDI architecture. In: Agents and Artificial Intelligence: Third International Conference, ICAART 2011, Rome, Italy, January, 28-30, 2011. Revised Selected Papers 3. pp. 176–191. Springer (2013)
31. Sardina, S., Padgham, L.: A BDI agent programming language with failure handling, declarative goals, and planning. Autonomous Agents and Multi-Agent Systems **23**(1), 18–70 (2011). https://doi.org/10.1007/s10458-010-9130-9, https://doi.org/10.1007/s10458-010-9130-9
32. Searle, J.: Making the Social World:The Structure of Human Civilization. Oxford University Press (2010)
33. Sichman, J., Noriega, P., Padget, J., Ossowski, S. (eds.): Coordination, Organizations, Institutions, and Norms in Agent Systems III, LNCS, vol. 4870. Springer (2007)
34. Silva, L.d., Meneguzzi, F., Logan, B.: BDI agent architectures: A survey. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. pp. 4914–4921 (7 2020). https://doi.org/10.24963/ijcai.2020/684
35. Singh, M.P.: Norms as a basis for governing sociotechnical systems. ACM Trans. Intell. Syst. Technol. **5**(1) (2014)
36. Söderström, T., Stoica, P.: System identification. Prentice Hall, Saddle River, New Jersey (1989)
37. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. Bradford, Cambridge (1998)
38. Szpunar, K.K., Spreng, R.N., Schacter, D.L.: A taxonomy of prospection: Introducing an organizational framework for future-oriented cognition. Proceedings of the National Academy of Sciences **111**(52), 18414–18421 (2014). https://doi.org/10.1073/pnas.1417144111, https://www.pnas.org/doi/abs/10.1073/pnas.1417144111
39. Yan, E., Nardin, L.G., Hübner, J.F., Boissier, O.: An agent-centric perspective on norm enforcement and sanctions. In: Cranefield, S., Nardin, L.G., Lloyd, N. (eds.) Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XVII. pp. 79–99. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-82039-7_6, https://arxiv.org/abs/2403.15128