Generalising Axelrod's Metanorms Game through the use of explicit domain-specific norms

Abira Sengupta[1], Stephen Cranefield[1], and Jeremy Pitt[2]

[1] University of Otago, Dunedin, New Zealand
[2] Imperial College London
abira.sengupta@postgrad.otago.ac.nz, stephen.cranefield@otago.ac.nz, j.pitt@imperial.ac.uk

Abstract. Achieving social order in societies of self-interested autonomous agents is a difficult problem due to lack of trust in the actions of others and the temptation to seek rewards at the expense of others. In human society, social norms play a strong role in fostering cooperative behaviour—as long as the value of cooperation and the cost of defection are understood by a large proportion of society. Prior work has shown the importance of both norms and metanorms (requiring punishment of defection) to produce and maintain norm-compliant behaviour in a society, e.g. as in Axelrod's approach of learning of individual behavioural characteristics of boldness and vengefulness. However, much of this work (including Axelrod's) uses simplified simulation scenarios in which norms are implicit in the code or are represented as simple bit strings, which limits the practical application of these methods for agents that interact across a range of real-world scenarios with complex norms. This work presents a generalisation of Axelrod's approach in which norms are explicitly represented and agents can choose their actions after performing *what-if* reasoning using a version of the event calculus that tracks the creation, fulfilment and violation of expectations. This approach allows agents to continually learn and apply their boldness and vengefulness parameters across multiple scenarios with differing norms. The approach is illustrated using Axelrod's scenario as well as a social dilemma from the behavioural game theory literature.

Keywords: Norms, Metanorms Game, Expectation Event Calculus

1 INTRODUCTION

The conflict between social benefit and an individual's self-interest is a central challenge in all social relationships as individuals may put their own interests ahead of those of the society as a whole, often leading to a suboptimal outcome for all—a situation known as a *social dilemma*.

Understanding how societies can solve this conflict and achieve cooperation toward the collective good is essential. In a fishing scenario, for example, it is profitable for a fisherman to catch as many fish as possible, but if everyone is selfishly doing the same thing, the fishery will eventually run out of fish.

The question of why people often do cooperate with others remains, despite the fact that individuals may benefit more from defecting than from cooperating. The classical game theory literature from the last few decades models social dilemmas using payoff matrices or trees and solution concepts such as the Nash equilibrium, while making the assumption that all agents are perfectly rational and well-informed. However, this becomes intractable to reason about for a large number of agents

Human society suggests that cooperation can occur due to internal and social motivations such as altruism, rational expectations (e.g. focal points), social choice mechanisms (e.g. voting and bargaining) and social norms [9].

One body of prior work has focused on the role of norms, providing evidence that social norms play an important role in fostering cooperation. Social norms imply that members of society should comply with prescribed behaviour while avoiding proscribed behaviours [15]. Bicchieri explains why agents adhere to social norms, claiming that a social norm emerges as a result of our expectations of others and beliefs about their expectations [4]. Axelrod's use of the norm and metanorms game illustrates how agents adopt normative behaviour after learning individual parameters of boldness and vengefulness, where their boldness represents their propensity to violate norms and vengefulness represents their inclination to punish others for violating norms [3]. However, his evolutionary study is based on an implicit representation of norms, which limits its practical use for agents that interact in a variety of real-world situations with a range of different norms.

In this work, we propose a generalisation of Axelrod's method where norms are represented explicitly and agents can choose their course of action after engaging in *what-if* reasoning to compare the normative outcomes of alternative actions. This approach is significant because it enables agents to continuously learn and apply their boldness and vengefulness parameters across a variety of scenarios with various norms.

The following is the structure of the paper. Axelrod's norms and metanorms games are discussed in Section 2. Section 3 emphasises the use of explicit norms to encode Axelrod's mechanism. Section 4 depicts the results of generalisation of Axelrod's norms and metanorms games, as well as the use of boldness and vengefulness in other scenarios. The prior event calculus models of norms are described in Section 5. Section 6 concludes the paper.

2 Background of Axelrod's model

Axelrod states that "the extent to how often a given type of action is a norm depends on just how often the action is taken and just how often someone is punished for not taking it". To understand how cooperation emerges from norms, he developed a game in which players learn the parameters of boldness and

vengefulness over generations of the population and can choose to deviate from the norms and metanorms, receiving punishment for their violations [3].

2.1 The norms game

Axelrod's norms game follows an evolutionary model in which successful agent strategies propagate over generations. A strategy is a pair of values representing the agent's boldness and vengefulness. Each agent has the option of defecting by violating a norm, and there is a chance of being observed by other agents with the probability S, which is drawn individually for each agent from a uniform distribution ranging . Each of the agents has two decisions to make (Figure 1(a)).

- Agents must decide whether to cooperate or defect based on their boldness value (B). A defecting agent (when $S < B$) receives a Temptation payoff (T = 3) while other agents receive a Hurt payoff (H = -1). If an agent decides to cooperate, no one's payoff will change as a result.
- If an agent observes others defecting (as determined by the S value), the agent decides whether to punish those defectors based on its vengefulness (V) (a probability of punishment). Punishers incur an enforcement cost (E = -2) every time they punish (P = -9) a defector.

Axelrod simulated the norms game five times with 100 generations of 20 agents[3]. Between generations, the utilities of each agent are used to evolve the population of agents. Agents with scores greater than the average population score plus one standard deviation are reproduced twice in a new generation. Agents with a score less than the average population score minus one standard deviation are not reproduced. Other agents are only reproduced once[4]. The initial values of B and V are chosen at random from a uniform distribution of eight values ranging from 0/7 to 7/7, with the numerator represented as a 3 bit string. During reproduction each bit has a 1% chance of being flipped as a mutation.

2.2 The Metanorms game

Axelrod found that norms alone were not sufficient to sustain norm compliance in society. He therefore introduced a *metanorm* to reinforce the practice of punishing defectors. The metanorms game includes punishment for those agents who do not punish defectors after observing them defect (Figure 1(b)). Metapunishers incur a meta-enforcement cost ($E' = -2$) every time they metapunish ($P' = -9$).

[3] Axelrod used five runs of a hundred generations to simulate the norms and metanorms games. However, we follow the recommendation of [7] and use 100 runs.

[4] Axelrod does not state how he maintains a fixed population size after applying these reproduction rules. We follow the approach of [7] involving random sampling when the new population is too large, and random replication when the population is too small.

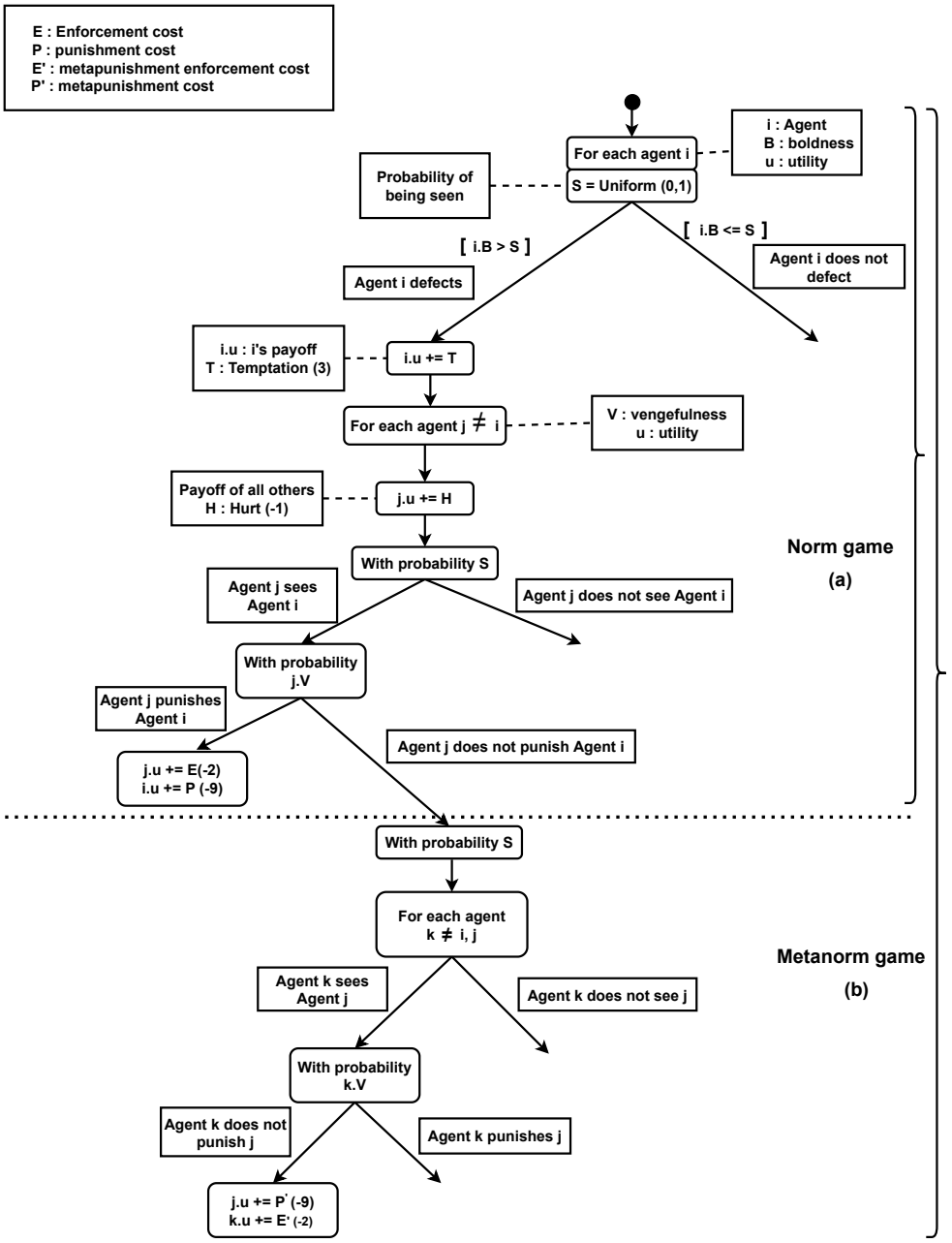

Fig. 1: (a) In Axelrod's **norms game**, agent i will defect if bold enough; otherwise, agent i will cooperate. Another agent j will punish i if the defection is observed and agent j is vengeful enough. (b) **The metanorms game** adds the possibility of metapunishment of agent j by another agent k. This occurs if j sees a defection from i, j does not punish i, this lack of punishment is observed by k and k is vengeful enough.

3 ENCODING AXELROD'S MECHANISM USING EXPLICIT NORMS

To generalise Axelrod's mechanism, we provide an explicit representation of norms and a mechanism that can compare alternative actions to determine which will lead to a norm violation. The expectation event calculus (EEC) [5], a discrete event calculus extension, provides this capability.

3.1 The Expectation Event Calculus

The event calculus (EC) consists of a set of predicates that are used to encode information about the occurrence of events and dynamic properties of the state of the world (known as fluents), as well as a set of axioms that interrelate these predicates [14]. This logical language supports various types of reasoning. In this work, we use it for *temporal projection*. This takes as input a narrative of events that are known to occur (expressed using $happensAt(E, T)$, where E is an event and T is a time point) and a domain-specific set of clauses defining the conditions under which events will *initiate* and *terminate* fluents (expressed using the predicates $initiates(E, F, T)$ and $terminates(E, F, T)$). The EC axioms are then used to infer what fluents hold at each time point. By default, fluents are assumed to have *inertia*, i. e. they hold until explicitly terminated by an event.

The EC, in general, assumes that time is dense, and time points are ordered using explicit '$<$' constraints. In this work, we use the *discrete event calculus* (DEC), which assumes that time points are discrete and identified by integers [11].

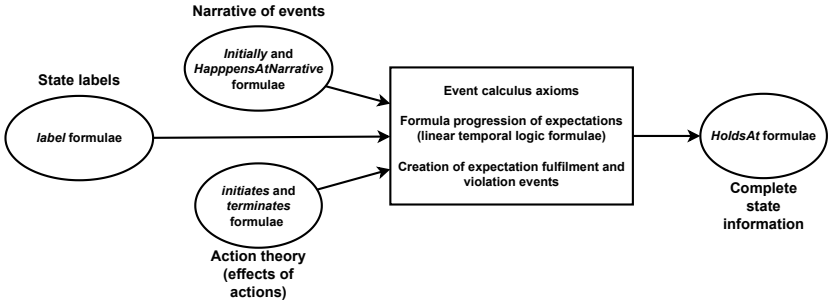

Fig. 2: Overview of reasoning in the expectation event calculus (EEC) [5]

The expectation event calculus (EEC) [5] is an extension of the DEC that includes the concepts of expectation, fulfilment, and violation. Expectations are constraints on the future, expressed in a form of linear temporal logic, that the

agent wishes to monitor. Expectations are free from inertia and instead are automatically *progressed* from one state to the next, which means they are partially evaluated and re-expressed in terms of the next time point. During progression, if they evaluate to true or false, a fulfilment or violation is generated.

Figure 2 illustrates temporal projection in the EEC. In addition to the standard features of the DEC, there are two special kinds of fluents: *exp_rule* and *exp*. A conditional rule to create expectations is expressed by an $exp\_rule(Cond, Exp)$ fluent. Here, *Cond* is a condition on the past and/or present, while *Exp* represents the future expectation. *Exp* will be expected if *Cond* holds, in which case an $exp(Exp)$ fluent is created. In our implementation of the EEC, the condition can test for fluents holding, the occurrence of events (expressed using $happ(E)$), the presence of a symbolic label L in a state (using the expression $@L$). Complex expressions involving conjunctions and linear temporal logic operators such as *next*, *eventually*, *always* and *never* can aso be used. Labels are associated with time points using $label(L, T)$ declarations, and are not required to be unique. To distinguish between basic events in the narrative and the inferred fulfilment and violation events, we use the predicate *happensAtNarrative* to declare the narrative events.

In contrast to the earlier approach of Cranefield [5], we represent fulfilments and violations as events rather than fluents, denoted $fulf(Cond, Exp, T, Res)$ and $viol(Cond, Exp, T, Res)$, where *Cond* and *Exp* are the condition and expectation of an expectation rule that was triggered at time T to create the expectation, and *Res* is the residual expectation (after being progressed zero or more times since its creation) at the time of its fulfilment or violation[5].

Our EEC implementation includes a *what-if* predicate that accepts two alternative event lists (E_1 and E_2) as arguments and infers the fluents that would hold and the events (including violation and fulfilment events) that would occur if each list of events were performed at the current time point. It returns the fluents and events that would occur if the events in E_1 are performed but not those in E_2, and those that would occur if the events in E_2 occur but not those in E_1. This can be used as a basic form of look-ahead to assist an agent in deciding between two actions, and especially in this work, to compare which (if any) of two events will cause expectation violations.

3.2 Modelling Axelrod's scenario with the EEC

We model time as a repeated cycle of steps and associate an EEC label with each step. We use the event calculus *initiates* and *terminates* clauses to define the effects of events that update an agent's S value, give payoff to an agent as the outcome of all agents' cooperate or defect actions, and punish and metapunish agents.

[5] There is also an extended version of the *exp* fluent with these four arguments—the version used in this paper has only the residual expectation as its argument.

We use the EEC within a simulation platform [6] that integrates Repast Simphony [12] with the EEC through queries to SWI Prolog. This includes an institutional model in which agents take on roles by asserting to the EEC narrative that certain institution-related events have occurred such as joining an institution and adding a role[6]. Each role has an associated set of conditional rules. A rule engine is run at the start of selected simulation steps where agents must choose an action and these role-specific rules recommend the actions that are relevant to the agent's current role based on queries to the EEC, e.g. to check the current step's label and the fluents that currently hold. Then the agent can run scenario-specific code to select one of the actions to perform[7].

In contrast to Axelrod's implicit representation of a norm and metanorm, our explicit representation of a norm implies that three norms are required. In the metanorms game, each action choice is governed by a norm. As there are three choice-points, there are three norms that we model using *exp_rule* fluents.

First-Order Norm

```
initially(
  exp_rule(member(A, society),
           never(happ(defect(A))))).
```

This `initially` clause creates an expectation rule (exp_rule) expressing the *first-order norm*, which states that no defection should occur for any agent who is a member of the society.

As the first-order norm described above is likely to be insufficient to motivate selfish agents to follow the norm and cooperate with others, a *second-order norm* is required.

Second-Order Norm

```
initially(
  exp_rule(and([sawViolation(B,A,R,_),
                pl(contains_term(defect(A), R))]),
           happ(punish(B,A)))).
```

The *second-order norm* states that if the *first-order norm* is violated by an agent, another agent who observes the violation should punish the first-order norm defector. The `pl` term in the rule's condition indicates a goal to be evaluated using Prolog.

This second-order norm is triggered by a *sawViolation* fluent, which is created when a violation of the first-order norm occurs and a defector is observed. The following *initiates* clause creates this fluent.

[6] https://github.com/maxant/rules

[7] At present we assume there are no more than two relevant actions.

```
initiates(viol(_,_,_,ResidualExp),
          sawViolation(B,A,ResidualExp,T),
          T) :-
    responsible(ResidualExp, A),
    agent(B),
    B \== A,
    holdsAt(s(A,S), T),
    random(R),
    R < S.
```

The condition for this clause first determines which agent is responsible for the unfulfilled expectation, then generates a possible observer different from the violator and compares the agent's S value with a random number to determine whether or not the violation has been observed.

In our application, the violated expectation will include an instantiation of one of the action terms `defect(A)`, `punish(A)` or `metapunish(A)`, and we can use these to identify the responsible agent. We therefore define the `responsible` predicate in Prolog as follows.

```
responsible(Expectation, A) :-
    contains_term(defect(A), Expectation).

responsible(Expectation, A) :-
    contains_term(punish(A,_), Expectation).

responsible(Expectation, A) :-
    contains_term(metapunish(A,_), Expectation).
```

To encourage the punishment of second-order norm violators, a *third-order norm*, is required.

Third-Order Norm

```
initially(
  exp_rule(and([sawViolation(B,A,R,_),
                pl(contains_term(punish(A,_), R))]),
           happ(metapunish(B,A)))).
```

According to this EEC rule, observer agents must punish the violators of the second-order norm when the violating agents fail to punish the first-order norm defector after observing their defection.

Figure 3(a) and (b) illustrate the differences between our implementation of the metanorms game with implicit and explicit norms. Figure 3(a) makes hard-coded action choices, following Axelrod's algorithm. However, in Figure 3(b), whenever there is an action choice to be made, the two action choices are compared using *what-if* reasoning that is informed by one of the three norms. If only one of the choices will cause norm violation, the agent's boldness parameter is used to decide whether the violating option is chosen.

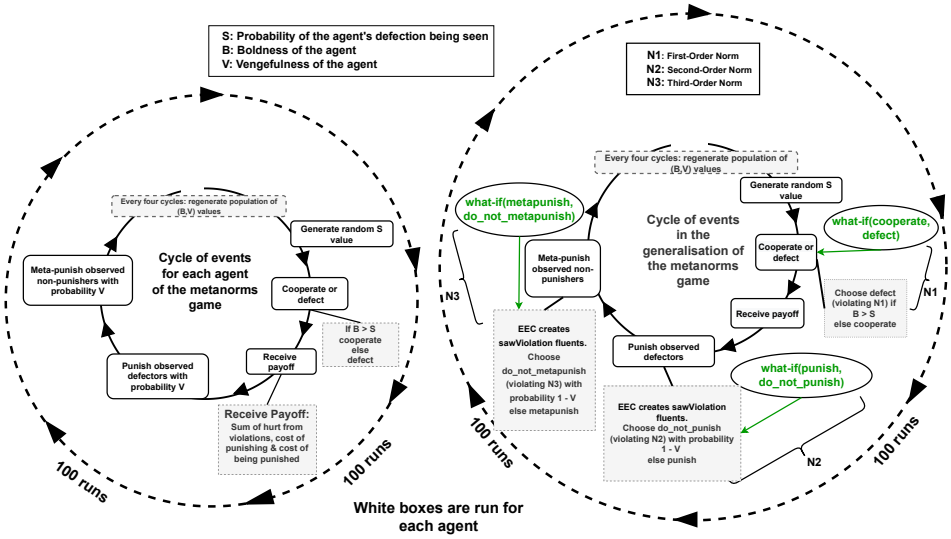

Fig. 3: The distinction between implicit and explicit metanorms. It depicts the cycle of events for Axelrod's metanorms game (a) in its original form with implicit norms, and (b) in our generalised form with explicit norms.

The box labelled Cooperate or defect indicates a time step in which an agent must decide whether to cooperate or defect based on whether it has a high boldness value (Figure 3(a)). In Figure 3(b), the first-order norm's *exp_rule* expressing the first-order will have been created in the initial time step and triggered once for each agent, resulting in *exp* fluents stating that each agent should never defect. Therefore, an agent can use the *what-if* mechanism to compare the outcomes of the two alternative actions, cooperation or defection. If the agent's boldness value exceeds S, the agent will violate the first-order norm by defecting; otherwise, the agent will cooperate.

The punishment step of the implicit norm simulation shows how agents punish each observed defector with a probability determined by the vengefulness parameter (Figure 3(a)). Whereas the explicit norm representation simulation cycle demonstrates second-order norm implementation within the punishment step and how the agent can use the *what-if* mechanism to punish or do not punish (Figure 3(b)).

If an agent violates with high boldness at the punishment step of the implicit norm simulation, then an agent with high vengefulness metapunishes the violator (Figure 3(a)). However, in our work, we demonstrate how the EEC uses the *third-order norm* at the meta-punishment step and how agents use the *what-if* mechanism to determine whether or not to meta-punish the violators (Figure 3(b)).

Table 1: Roles and their possible actions

Step	*Role*	*Possible actions*
cooperate or defect	temptation role	cooperate or defect
punishment	possible punisher role	punish or do not punish
metapunishment	possible punisher role	meta-punish or do not meta-punish

Figure 4 depicts in more detail our use of explicit norms with agents that are aware of norm violations. We have three norms that represent rules in different time steps, and they triggered and created expectations. The EEC *initially* clauses generated the *exp_ rule* fluents for the first-order norm (N_1), second-order norm (N_2), and third-order norm (N_3). Each agent has two roles: *temptation role* and *possible punisher role*. Table 1 shows what actions an agent can take in the simulation when assigned to a specific role for each step. The *temptation role* specifies that an agent can choose to cooperate or defect at the *cooperate or defect* step. While at the *punishment* step an agent with the *possible punisher role* can choose to punish or do not punish, and an agent with the *possible punisher role* can choose to metapunish or not to metapunish at the *metapunishment* step. At the initial time step of the simulation, both roles (*temptation role* and *possible punisher role*) are activated for each agent.

The EEC *what-if* predicate is used to consider two options: cooperate or defect, punish or do not punish, metapunish or do not metapunish (depending on the current step in the simulation cycle), and determine whether one rule produces a violation while the other does not. The non-violating option is then chosen (or a random choice if there is no violation). If both options result in a violation, the cost of each violation is calculated (using domain-specific knowledge) and the less costly option is chosen. If the costs are the same, a random selection is made.

At the simulation's final step, regenerate_population successful agents are replicated and mutated to form a new generation of the same size [7]. In this simulation, folded outlined arrows represent iteration: one for the three norms and their corresponding expectations within one generation, and the other for 100 generations of simulation.

4 Results

A. Experiment 1: Generalisation of Axelrod's metanorms game The simulation of the experiment depicts what happens when agents violate the second-order norm and are metapunished by vengeful agents adhering to the *third-order norm*. Figure 5(a) shows a scatter plot representation demonstrating that boldness is always low and vengefulness ranges from high to average. We use twenty agents, 100 generations, and 100 runs to simulate Axelrod's generalisation

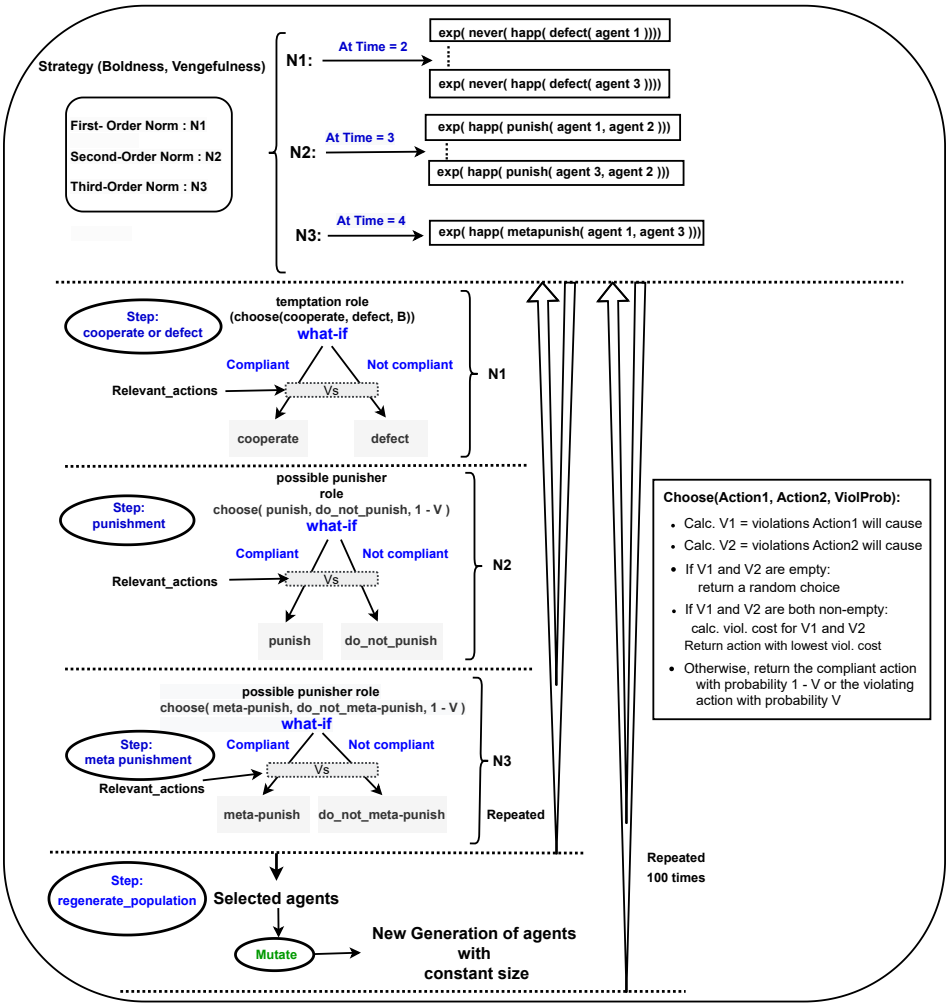

Fig. 4: The generalisation of Axelrod's approach in which norms are explicitly represented and agents can choose their actions based on *what-if* reasoning using expectation event calculus, which tracks the creation, fulfilment, and violation of expectations. Agents 1, 2, and 3 are expected to never defect under the first-order norm. We can assume that agent 2 is a defector. Then, according to the second-order norm, agents 1 and 3 are expected to punish agent 2. Then, according to the third-order norm, if agent 1 notices that agent 3 did not punish agent 2, agent 1 will punish agent 3.

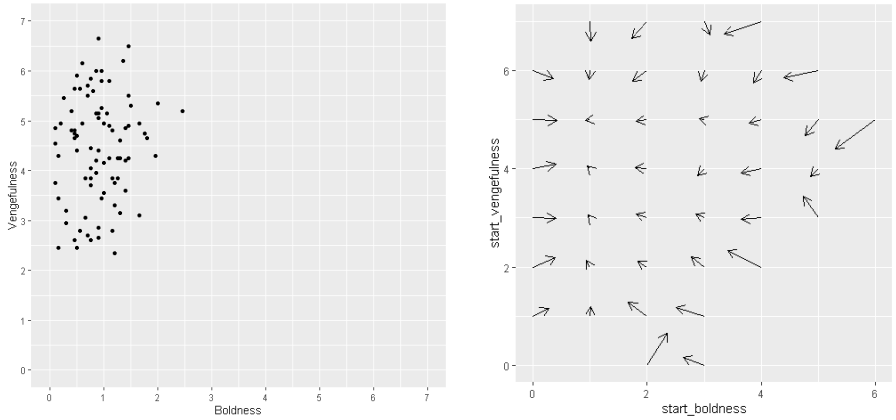

Fig. 5: (a) **Scatter plot** of mean boldness along x-axis wise and mean vengefulness along y-axis wise of generalisation of Axelrod's study with all three norms. (b) **Vector plot representation** of mean boldness and vengefulness.

of explicit norms. The scatter plots generated by the mean value of boldness and vengefulness in each of 100 runs.

Figure 5(b) depicts the vector representation of the same data set[8]. Vectors show how boldness and vengefulness change across generations in the population. The results show that *what-if* reasoning with explicit norms causes the first-order norm to be largely upheld in the society due to low boldness being maintained.

However, compared to Axelrod's results with implicit norms, there remains some moderately low vengefulness values. This may be explained by a limitation of the rule engine we use: only a single predetermined string can be returned as the result of a rule, e.g. "punish". Thus, when the rule's condition compares an agent's vengefulness with a random number, the outcome is either to punish all or none of the observed violators. In contrast, in Axelrod's algorithm, each individual punishment is the result of a different randomise decision. Further investigation is needed to determine whether this explains the ability for agents with lower vengefulness to remain in the population.

B. Experiment 2: Using the boldness and vengefulness in another scenario [10] introduced a scenario that we refer to as the plain-plateau scenario in our previous work [13]. The scenario depicts a society in which people have the option of living on a river plain with easy access to water, otherwise, they can live on a plateau. Flooding is a risk for river-plain residents. When the government has complete discretionary power, it is in the government's best interests

[8] Populations with similar average levels of boldness and vengefulness are grouped together to create each vector. The end point of each arrow shows the average levels of these features one generation later.

to compensate citizens whose homes have been flooded by taxing citizens who live on the plateau, creating a prisoner's dilemma situation. In previous work, we experimented with the use of social norm-based expectations to achieve coordination where citizen agents are hard-coded to prefer actions that will result in no violation.

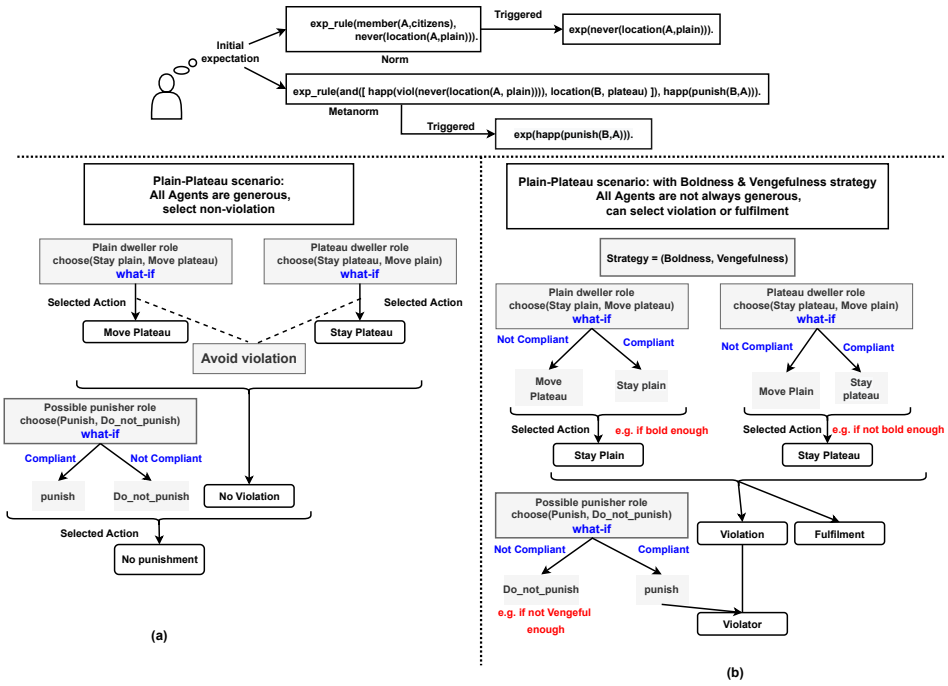

Fig. 6: (a) **The plain-plateau scenario** in which agents are hard-coded to always choose the non-violating actions. (b) **The plain-plateau scenario** as an application of our generalised metanorms game.

Figure 6(a) illustrates this prior work. Each agent has either a plain dweller or a plateau dweller role, and in each simulation cycle there are two choices: agents can stay on the plateau or move to the plain. In this scenario, we assume there exists a norm that no one should live in the plain and a metanorm stating that plateau dwellers should punish those who live on the plain.

Figure 6(b) illustrates the application of our generalisation of the metanorms game to this scenario[9]. We import boldness and vengefulness parameters from a run of Experiment 1. This simulates an agent finding itself in a new scenario after having already evolved its personality with respect to norms, and illustrates the generality of our approach using explicit norms and *what-if* reasoning. In the

[9] Currently, we do not include a level 3 norm for the plain-plateau scenario.

plain-dweller role, the EEC *what-if* predicate is used to consider two options: move to the plateau or stay in the plain; similarly, in the plateau-dweller role, the *what-if* mechanism is used to consider either move plain or stay plateau. When agents with high boldness in the plain-dweller role choose to stay in the plain, they violate the norm. When other vengeful agents observe violators, they punish them, unless insufficient vengefulness causes them to violate the metanorm.

5 Prior event calculus models of norms

This section of the paper reviews some research on the use of event calculus in autonomous agent reasoning to examine the effects of norms.

Alrawagfeh [1] suggests formalising prohibition and obligation norms using event calculus and offers a method for BDI agents to reason about their behaviour at runtime while taking into account the norms in effect at the time and previous actions. Norms are represented by EC rules that initiate fluents with special meanings. The introduced fluents represent punishments for breaking a prohibition norm or failing to fulfil obligation norms, or the rewards for fulfilling obligation norms. The normative reasoning strategy assists agents in selecting the most profitable plan by temporarily asserting to the event calculus the actions that each plan would generate and considering the punishments and/or rewards it would trigger.

In Alrawagfeh's work, norms cannot be changed dynamically without changing the event calculus rule base, because they are defined by EC initiates clauses. In contrast, in our approach, EC rules can be instantiated automatically from *exp_rule* fluents, which can be changed dynamically by events.

Alrawagfeh has no representation of active norms, violations or fulfilments: only punishments and rewards. In our work, expectation creation, fulfilment, and violation are represented as events, and the *what-if* predicate compares alternative events to track expectation creation, fulfilment, and violation. We do not assume that rewards and/or punishments will always follow violations and fulfilments; these could be defined by separate *exp_rules* or EC initiates clauses.

Hashmi et al. [8] propose a number of new EC predicates to allow them to model different types of obligation that occur in legal norms. In particular, they introduce a *deontically holds at* predicate that ensures an obligation enters into force at the same time that the triggering event occurs. In contrast, our approach does not necessitate the introduction of a new type of EC predicate in order to initiate a deontic predicate. An *exp_rule* or an expectation can be created with a standard initiates clause and an *exp* fluent is created by an *exp_rule* in the state where the condition of the rule becomes true. However, the EEC includes additional axioms to deal with the progression of expectations.

Alrawagfeh and Hashmi et al. both use standard EC, whereas we use discrete EC because this work involves discrete time simulations.

6 Conclusion and future work

In previous work, we used the EEC *what-if* mechanism for choosing actions in the presence of expectations how agents adhere to norms and choose actions using the *what-if* mechanism. However, we assumed that all agents are hard-coded to avoid expectation-violating actions.

The current work builds on this previous work to remove this assumption, but it also makes the following significant standalone contribution. It generalises Axelrod's study to use explicitly represented norms. This allows the metanorms game to be used across multiple scenarios.

Applying our generalised version of Axelrod's metanorms game to varying scenarios will require changing the mechanism for evolving boldness and vengefulness parameters. Strategy evolution through population regeneration is not realistic for agents that continually evolve their boldness and vengefulness as they move between different scenarios. Therefore, in future work we will investigate the use of a pairwise comparison approach where an agent may adopt another agent's strategy based on a comparison of their respective fitnesses, e.g. by using the Fermi process [2].

References

1. Alrawagfeh, W.: Norm representation and reasoning: a formalization in event calculus. In: International Conference on Principles and Practice of Multi-Agent Systems. pp. 5–20. Springer (2013)
2. Altrock, P.M., Traulsen, A.: Fixation times in evolutionary games under weak selection. New Journal of Physics **11**(1), 013012 (2009)
3. Axelrod, R.: An evolutionary approach to norms. American Political Science Review **80**(4), 1095–1111 (1986)
4. Bicchieri, C.: The grammar of society: The nature and dynamics of social norms. Cambridge University Press (2005)
5. Cranefield, S.: Agents and expectations. In: International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems. pp. 234–255. Springer (2013)
6. Cranefield, S., Clark-Younger, H., Hay, G.: A collective action simulation platform. In: Multi-Agent-Based Simulation XX: 20th International Workshop, MABS 2019, Montreal, QC, Canada, May 13, 2019, Revised Selected Papers 20. pp. 69–80. Springer (2020)
7. Galan, J.M., Izquierdo, L.R.: Appearances can be deceiving: Lessons learned reimplementing Axelrod's evolutionary approach to norms. Journal of Artificial Societies and Social Simulation **8**(3) (2005)
8. Hashmi, M., Governatori, G., Wynn, M.T.: Modeling obligations with event-calculus. In: Rules on the Web. From Theory to Applications: 8th International Symposium, RuleML 2014, Co-located with the 21st European Conference on Artificial Intelligence, ECAI 2014, Prague, Czech Republic, August 18-20, 2014. pp. 296–310. Springer (2014)
9. Holzinger, K.: The problems of collective action: A new approach. MPI Collective Goods Preprint No. 2003/2, SSRN (2003), doi:10.2139/ssrn.399140
10. Klein, D.B.: The microfoundations of rules vs. discretion. Constitutional Political Economy **1**(3), 1–19 (1990)
11. Mueller, E.T.: Commonsense Reasoning. Morgan Kaufmann (2006)
12. North, M.J., Collier, N.T., Ozik, J., Tatara, E.R., Macal, C.M., Bragen, M., Sydelko, P.: Complex adaptive systems modeling with Repast Simphony. Complex Adaptive Systems Modeling **1**(1), 3 (2013)
13. Sengupta, A., Cranefield, S., Pitt, J.: Solving social dilemmas by reasoning about expectations. In: Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XIV. pp. 143–159. Springer (2022)
14. Shanahan, M.: The event calculus explained. In: Artificial Intelligence Today, pp. 409–430. Springer (1999)
15. Thøgersen, J.: Social norms and cooperation in real-life social dilemmas. Journal of Economic Psychology **29**(4), 458–472 (2008)