University of Glasgow | School of Computing Science

Investigating Computational Responsibility

William Wallis (2025138)

April 20, 2017

ABSTRACT

A growing area of research lies in the formalism of human traits into computational representations. Formalisms of trust, comfort, and other traits have brought new techniques to human-computer interaction, information security, sociotechnical modelling and other fields. Recently, interest has grown around the trait of responsibility — however, constructs of responsibility have not yet taken an computational form. In this paper, a proof-of-concept computational responsibility formalism is produced, and evaluated as a task scheduling mechanism.

1. INTRODUCTION

1.1 Background

Computational formalisms of human traits are a growing field of research, with applications in a number of different areas. These algorithms make computers more human-like; for that reason, they are referred to here as "anthropomorphic algorithms"[1]. These algorithms have a diverse range of applications:

- Human-Computer Interaction can make use of behavioural metaphors to relay complicated internal states to a user. Storer et al. [17] demonstrated methods by which a mobile device might dissuade certain user actions by expressing its "discomfort" or lack of "trust" in its interaction with the user.
- Information Security can make use of behavioural metaphors in order to increase difficulty of access when negative interactions are encountered. A system might allow access on a graded scale, dependant on internal states of trust, comfort, and/or confidence.
- Theoretical advancements in smart city technology might increase a city's resilience by integrating notions of responsibility into public services and the environment on a community scale, as well as in pervasive computing[20].
- Implementation of human behaviours as anthropomorphic algorithms lends itself to directing modelled agent behaviour in a similar way. Sociotechnical systems modelling enhances agent behaviour using these formalisms of human behaviour[2].[10].

[1]A similar term, "human-like computing", has also risen in popularity lately. Human-like computing does not strictly focus on the implementation of formalisms of human traits, however, which is the area of interest for this paper.

[2]A formalism here refers to a well-structured, agreed-upon set of definitions

These application areas show the broad possibilities for anthropomorphic algorithms research. These are advanced fields which continue to see improvements when anthropomorphic algorithms research is applied, as this alternative approach enables new avenues of research to be undertaken.

1.2 Motivation

Despite the activity in this area, there is considerable opportunity to explore the formalisation of alternative social traits as anthropomorphic algorithms (as is demonstrated during the background survey in section 2). The lack of development of anthropomorphic algorithm springs from a range of related factors. For example, existing approaches to anthropomorphic algorithms make a trade-off between utility in a specific problem context or setting; and fidelity to the underlying social construct. This results in two philosophical approaches to anthropomorphic algorithm design:

Type 1: Trait as a useful metaphor
These models tend to be inaccurate with regards the social science surrounding the trait that they model. However, they make a trade-off between this accuracy and the model's utility. For example, the notion of trust as a metaphor for a type of behaviour might be useful in information security research, but what matters in the formalisms implemented for this research is the formalism's utility in information security — *not* whether the formalism accurately represents human trust. The Eigentrust formalism[6] is an example of trust used as a description of desired behaviour in information security, which relied on no social science research to inform its definition of "trust".

Type 2: Modelling social science directly
These models attempt to accurately model the traits they concern. This can be useful for fields such as sociotechnical modelling, as well as social sciences research. There are also interesting applications for these models in interaction study: making interfaces interact with users in a human-like way, and representing the states of these traits to the users, are valuable research areas which are more applicable to these type-2 formalisms than to type-1 formalisms. Marsh's formalism of trust[10] is an example of a trust formalism which is not designed for a very specific application, but rather simulates the social science research it finds its foundation in.

In reality, most formalisms and their implementations lie somewhere on the spectrum that these two types define. In either case, research into anthropomorphic algorithms is strongly interdisciplinary in nature: it requires a research

team to understand the nuances of sciences as well as social sciences, and sometimes even humanities. This often causes divergence in the philosophies of research agendas in different disciplines. Different formalisms are useful to different researchers, for different modelling applications, with different philosophies. This also results in a lack of consistent representations of anthropic traits[3].

The lack of broad application is therefore unsurprising. These factors compound to ensure that the number of traits formalised and researched is very small. The largest degree of research is easily conducted in the field of trust; other traits, such as comfort, have also recently been attempted [9].

1.3 Contribution and Research Questions

Recently, some interest has been shown in research pertaining to modelling and formalising *responsibility*[1, 14, 12]. These often feature modelling the delegation and discharge of a task, which can be constrained by certain factors — such as availability of necessary resources, or the discharge of other responsibilities which another depends on. A wide array of responsibility modelling techniques exist already which might be useful in developing an algorithmic model of responsibility.

In drawing on this available literature, various philosophical approaches to the definition of responsibility present themselves. Logical models of social trust exist which could be turned into a formalism of responsibility, delegating responsibilities to agents who are deemed "responsible" if they can be trusted. Indeed, similar methods are employed to infer trustworthiness from scores of reputation [6]. A model of responsibility might be more than simple task allocation, however. Some logical models of responsibility attempt to model ethically responsible decisions [1].

Deontic logic, for example, is an attempt to create a logic which was suitable for the calculation of whether an agent was obliged to ensure certain goals were met, or perform certain tasks, from a philosophical standpoint. Another angle might be to perceive a model of responsibility as something which might allow a responsible agent in a sociotechnical context to *choose* responsibilities to discharge, rather than blindly executing tasks they are provided with in a model which simply delegates tasks.

An anthropomorphic algorithm for "computational responsibility" would direct an agent's behaviour; this might favour the discharge of more important tasks earlier in the agent's timeline. A suitable definition of responsibility for these purposes might be that responsibility is a property of an agent which directs it behaviour toward prioritising a task's importance. Such a model of responsibility does not exist in in algorithmic format.

The contribution of the work presented here is to create a formalism of responsibility suitable for the delegation and selective discharge of responsibilities, as well as the calculation of how responsible an agent is "perceived" to be, according to the above definition of responsibility. This task scheduling-oriented approach permits future study, such as eventual specific applications in areas such as RPC, as work which builds on this solid foundation. As such a formalism does not currently exist, in a format suitable for directing agent behaviour, the presented work will be a proof-of-concept formalism suitable for this more general purpose.

In many contexts, an agent's behaviour is often informed via a feedback loop. It is important, therefore, to allow an agent to learn better "responsible" behaviour over time, through an analysis of its sociotechnical environment and other factors. Anthropic traits are most useful when they account for both introspection — so as to direct an agent's own behaviour based on the formalised trait — and extrospection — so as to judge and learn from that trait in other agents.

To achieve these goals, two research questions were formulated:

1. How can a computational formalism of responsibility direct the decisions made by an intelligent agent?
2. How can an intelligent agent assume the consequences of actions it makes, the decisions other agents make, and its general environment, so as to direct its interpretation of responsibility?

1.4 Outline

This paper is structured as follows. Section 2 explores the problem domain, and contextualises the research questions which this paper will address. Section 3 discusses the design decisions inspired by this related work, and details a suitable model for computational responsibility. Following this, section 4 presents an evaluation of the formalism using responsibility as a metaphor in task scheduling. Section 5 explores potential future work building on this proof-of-concept computational responsibility model. Section 6 summarises the contributions of the model to the anthropomorphic algorithms field, as well as providing a summary of the formalism's efficacy as a responsibility model.

2. RELATED WORK

A broad range of literature must be reviewed to properly understand the research at hand, due to the problem's unusual nature. Particularly, this background literature survey will focus on three areas: popular algorithmic trust models, broader sociotechnical systems research, and relevant philosophical literature.

2.1 Trust Modelling

Ordinarily, when constructing a new anthropomorphic algorithm, one would draw on literature regarding other anthropomorphic algorithms which formalise the same trait with the same intended purpose. However, no such formalism exists for responsibility modelling; therefore, insights from early trust modelling may provide useful information on how to proceed, given that responsibility formalism now is in a similar stage to early trust formalism. Trust is particularly appropriate as a comparative trait to responsibility, as the two traits have many similarities. Trust and responsibility both concern tasks, and their execution, within certain formalisms, yet require different perspectives on this execution. These similarities will be explored further as they appear in the explored trust literature.

2.1.1 Marsh [10]

The seminal anthropomorphic algorithm for trust is found in Marsh's 1994 formalism[10]. In this paper, a formalism

[3]"Anthropic traits" are behavioural traits of human beings. "Anthropomorphic algorithms" are approximations of these traits.

is described which has a number of useful qualities: it is modelled on a per-agent basis as opposed to calculating trust across a group of agents, has foundations in social sciences, and is largely algebraic, staying clear of modelling via logic.

Of particular interest is Marsh's separation of three different degrees of detail in trust judgement. Marsh identifies that human agents have:

"Basic Trust": A basic degree or weighting which applies to their trust — it would not be uncommon to assert that "person X is very *trusting*".

"General Trust": An assertion that "person X is trusting, but *doesn't trust person Y*" — the additional parameter of a second agent in this trust calculation generates more detail as to the nature of the judgement of trust.

"Specific Trust": This identifies an agent's degree of trust in another with regards a specific task (which could be considered "person X's degree of trust in person Y *in doing task* α). As with "General Trust", the added parameter generates another degree of detail in the trust judgement.

These separations are useful in a few key ways. One is that it identifies some parameters in the model of trust: at the very least, Marsh's formalism of trust requires an agent, two agents, or two agents and a task or action in order to calculate a degree of trust. This is useful when generating a model of responsibility, as trust and responsibility share some common features: like trust, responsibility usually concerns two agents, and a task that one agent is responsible for. Unlike trust, responsibility modelling contains hierarchies: one agent might be considered the authoritative figure, which *delegates* a task to another. For the purposes of this paper, the latter agent will be referred to as a "delegee", and the former the "authority". The similarities in the parameters of the formalism affirm the notion that responsibility and trust are similar in concept, and also serve as a starting point for imagining what an algorithmic formalism of responsibility might be like.

Another useful insight from these separations would be that, when considering responsibility, judging how responsible an agent is can be done from the same three vantage points. Basic responsibility would colloquially be "benefit of the doubt", general responsibility would be how responsible an agent is in all modelled capacities, and specific responsibility would be another agent's calculated responsibility with regards some task, resource, or other modelled subject of responsibility. Not only does this also affirm trust and responsibility's similarities, but it helps in providing further structure to the new formalism.

A final important property of Marsh's formalism is its graded nature. This model does not produce a boolean indicator of whether to trust or not; rather, it opts to calculate a number between 0 and 1 of the *degree* of trust one agent ought to have in another. This feature allows for as granular a model of trust as any given application requires, and allows for nuanced comparison between agents.

2.1.2 Castelfranchi & Falcone [4]

Another useful formalism of trust to consider comes from Castelfranchi & Falcone[4] . While this formalism has logical foundations, its popularity makes it an interesting comparison to Marsh's formalism.

In contrast to Marsh's formalism, Castelfranchi & Falcone's model is graded only in more complex forms. At its root, Castelfranchi & Falcone's model is a logical formalism built on boolean calculations of goal and belief satisfaction. However, Castelfranchi & Falcone also segregate different aspects of trust in their formalism: different elements of the basic formalism represent competence, disposition, and dependence. They define competence as one agent's belief and will that another agent can successfully achieve a goal by a certain action, disposition as the belief and will that the other agent is willing to perform an action to achieve a goal, and dependence as the effective delegation of a task for the purpose of achieving a goal (expressed via logical statements).

Curiously, the Castelfranchi & Falcone model of trust contains all of the same parameters as Marsh's, plus a fourth: the goal to be achieved by an action. One may take the position that a goal is also important in modelling responsibility — but it is not immediately apparent that it is necessary to include both. Therefore, Castelfranchi & Falcone uncovers the need when building a formalism of a trait to make choices as to the formalism's philosophy.

In particular, two design differences between Marsh's model and that of Castelfranchi & Falcone are immediately apparent: where Marsh separates his calculation of trust into a basic/general/specific granularity, Castelfranchi & Falcone separate trust's different constituent parts, each of which must be satisfied for trust to be present. Another philosophical difference between the two is the assertion as to whether the goal of an action factors into one's trust. These philosophical decisions can make concrete, important differences to a formalism's constitution — which is plain to see when considering that even the parameters of the model differ, something fundamental to the formalism's definition of trust. Therefore, these choices as to the approach to responsibility are important to note in the formalism produced during this paper.

2.1.3 Eigentrust [6]

Eigentrust[6] is a formalism which takes a notably different approach to Marsh and Castelfranchi & Falcone's formalisms: rather than attempting to model human trust accurately, it models trust as a metaphor for the truly desired behaviour.

Calculations of trust in Eigentrust have their foundations in eBay's model for reputation: star ratings based on a summation of satisfaction scores. In particular:

$$s(i,j) = sat(i,j) - unsat(i,j)$$

... where *sat* is the number of satisfactory interactions between two agents and *unsat* the unsatisfactory ones, represents a "local trust value" that an agent i has in another agent j. Through some linear algebra, these local trust values are accumulated and gradually turned into a global score of responsibility, accounting for all agents' opinions of each other. This global score could also be considered to be an agent's *reputation* — in this way, Eigentrust generalises trust as a certain application of a reputation formalism, and bootstraps its own formalism on another trait.

While Eigentrust has proven particularly effective in its intended domain, it is a formalism designed expressly for the

purposes of network and information security. Eigentrust achieves this by a number of philosophical differences to Marsh and Castelfranchi & Falcone's respective formalisms:

- Eigentrust operates in a distributed way. Unlike the local scoring system employed by Marsh and Castelfranchi & Falcone's formalisms, Eigentrust has all agents report their local trust scores, so as to create a distributed ledger of collectively agreed-upon trust scores.
- Eigentrust does not model trust directly, nor does it claim to model it accurately — unlike Castelfranchi & Falcone's formalism, which is strictly intended as a model of human behaviour, and Marsh's, which simulates it, Eigentrust uses "trust" as a loose description of an agent's behaviour.
- Eigentrust is not concerned with the cause of satisfactory or unsatisfactory interactions; it focuses entirely on accumulated positive/negative scores. In this way, Eigentrust somewhat models Marsh's "general trust", but makes no assertions as to trust's composition or how to reason about it.

In these respects, Eigentrust represents an interesting alternative end of the spectrum between anthropomorphic algorithms which treat their trait as a metaphor, or as a social behaviour to accurately simulate. While Eigentrust does not represent a very useful foundation for our responsibility formalism, it does highlight two things:

1. The responsibility formalism required to test the research questions should be more similar to Marsh and Castelfranchi & Falcone's formalisms than Eigentrust: there is no specific use case to design for, so designing with a trait as a metaphor would not be appropriate.
2. Should it be necessary, a trait's formalism could be bootstrapped using a pre-existing formalism of another trait. Whether this is a suitable way to answer the research questions above is harder to address; the possibility should therefore be considered.

2.1.4 FIRE [5]

FIRE[5] is another trust modelling system which provides a focus on being able to judge trust using information from many different sources. For example, it treats direct experience information in a different way to information collected by third parties, unlike other trust formalisms, which concern themselves with only a single source or type of information to calculate using. It is also a model which considers multiple different traits: it incorporates reputation information into its judgement of trust. In part, FIRE is able to do this because it segregates different information sources into different measurements, which are tabulated into a score after their measurement.

FIRE's inclusion of information from multiple sources paints other trust formalisms in a slightly different light. However, this feature is not necessary for all trust scenarios: the decision to trust (or not trust) is made with different amounts of information for agents in different simulations. One can imagine a two-agent simulation, where information about each agent's interactions would be assessed by exactly one source — the other agent, which judges the former's reliability. It is plain to see that FIRE is designed to be a formalism treating traits as a metaphor, similarly to Eigentrust.

This design decision makes FIRE an unlikely candidate as a foundation of a responsibility formalism, as its specific application area — sociotechnical models with multiple types of information to consider — is more complex than is necessary for a proof-of-concept responsibility formalism, and would be more complicated than the research questions posed require. However, the design choice it raises regarding different types of information, and the nature of the information which is being reviewed when calculating the responsibility score, is an important one.

2.2 Sociotechnical Systems

While these formalisms are useful to consider in isolation, their application within the realm of sociotechnical systems is important to their design. Moreover, the nature of responsibilities is touched within the broader sociotechnical systems area of responsibility modelling: research on the delegation and discharge of responsibilities, and how to reason about them.

2.2.1 Sommerville [8, 14, 13]

Ian Sommerville was a prolific writer in the field of sociotechnical systems, who was responsible for much of the current literature on responsibility modelling[8, 14, 13].Sommerville's responsibility modelling systems often happened to be graphical[8], a paradigm for computational responsibility which may prove hard to convert to an anthropomorphic algorithm. This is because graphical representations of system states do not naturally present themselves as a numerically analysable format; rather, graphical presentations are useful for exposing sociotechnical system state. This feature of sociotechnical modelling, and particularly responsibility modelling, is useful for risk and impact analysis[19].

Sommerville's writing, however, presents a wealth of interesting insights which may be useful in understanding the context of an anthropomorphic algorithm, and understanding some possible choices as to the formalism's philosophy.

In particular, Sommerville notes an interesting distinction as to two different sorts of responsibilities: "consequential" and "causal" responsibilities. They can be thought of as the difference between a responsibility for a current state — the result of a previously discharged responsibility — and responsibility for producing a future state — a change to a current state that an agent is responsible for bringing about in the future.

Sommerville's separation clarifies a potential avenue by which one might create a computational responsibility formalism. Namely, an agent's degree of responsibility might be associated with the state changes they have brought about in the past, and perhaps a prediction of the chance that an agent would discharge a causal responsibility successfully at some future time. This approach, verified by its appearance in existing sociotechnical systems literature, may be appropriate for creating a computational responsibility formalism, though its foundations exist in a less useful graphical representation.

2.2.2 Lock & Storer [16]

While describing a graphical responsibility modelling format for the InDeED project, Lock & Storer provide some useful foundations for other models of responsibility in a technological report[16] .

Most interestingly, Lock & Storer provide a useful defini-

tion of a responsibility:

> A duty, held by some agent, to achieve, maintain or avoid some given state, subject to conformance with organisational, social and cultural norms.[16]
> ...
> Responsibilities are the duties to be discharged by agents as described...

Numerous things in this definition are useful. Similarly to the work done by Marsh and Castelfranchi & Falcone, the definition provides an insight into some possible fundamental parameters of a responsibility:

A duty: responsibilities can be considered as some action an agent is *obliged* to conduct.

achieve... some given state: the obligation which a duty is defined by can be described as a change of *state*. Lock & Storer note that this state change can have different modes: it can be achieved, but also maintained or avoided. It is worth noting that this very general description of responsibility indicates that the graphical formalism designed for the InDeED project was somewhat socially accurate.

comformance with... norms: agents can be delegated responsibilities in most useful formalisms of responsibility — Lock & Storer note that these responsibilities must not conflict with pre-existing beliefs about the responsibilities that an agent holds.

Unlike Marsh or Castelfranchi & Falcone's more mathematical definitions of trust, this semantic definition lends itself nicely to conversion to some responsibility model; partly because it already summarises the responsibility trait, but also because it describes an intuitive social definition of responsibility while remaining very simple. Therefore, a responsibility formalism which somehow extended this model, while retaining the power of the anthropomorphic algorithms described earlier may prove useful in answering our research question.

Another useful definition from Lock & Storer's report presents itself when discussing the structure of a responsibility in their model:

> Responsibilities may be composed of other responsibilities.

This composibility is an interesting property, which may also imply that some responsibilities can be broken down into a form of sub-responsibility. This property would be useful to remember when modelling responsibilities in the desired formalism. It also implies that there may be a sort of "atomic" responsibility, that more complicated responsibilities are composed of, if decomposition is permitted as well as composition.

2.3 Philosophical Literature

As responsibility is a social trait, and previous literature has indicated that it may be most appropriate to model it as such, input from literature in the humanities may shed light on how to consider and approach responsibility as a trait accurately. As time allowed for this paper was somewhat limited, a more complicated sociologically/psychologically accurate model was not attempted. The decision was made to instead create a proof-of-concept model which encapsulated much of the existing sociotechnical systems literature, and verifying it using philosophical literature on the subject.

2.3.1 Scanlon [11]

Thomas Scanlon — a philosopher who specialises in Analytic Philosophy — writes on responsibility in the essay "Justice, Responsibility, and the Demands of Equality", defining terms which bear striking resemblance to Sommerville's "consequential" and "causal" responsibilities.

One term — "attributive" responsibility — is defined by Scanlon as being:

> What a person sees as a reason for acting, thinking, or feeling a certain way[11]

This might be generalised as being the responsibility for future and current action, as attributive responsibilities would clearly be the influencing responsibility of a future action. If future actions discharge the responsibilities an agent possesses at a given time, then Scanlon's "attributive" responsibilities in effect mirror Sommerville's "causal" responsibilities.

Scanlon goes on to define a second term, "substantive" responsibility, which can be described loosely as a responsibility for an action fixed in the past. Given the sociotechnical perspective of responsibility being a change of sociotechnical state, Scanlon's "substantive" responsibilities also seem to mirror a term of Sommerville's, namely his "consequential" responsibilities.

Given the convergence of this perspective on responsibility — that is, the separation of past state change and anticipated future state change — Scanlon's writing seems to affirm Sommerville's own construction of types of responsibility. Clarity on how this distinction is useful is found in work from another philosopher, P.F. Strawson.

2.3.2 Strawson [18]

P.F. Strawson is a philosopher with work on philosophy of language and of mind — one particular essay, "On Freedom and Resentment"[18], presents an argument that determinism shouldn't affect how we perceive human factors such as responsibility or trust.

Strawson argues that these concepts are fundamentally relational, rather than having any inherent definition. The conclusion of this is that trust, responsibility, and other human traits have subjective properties and cannot be calculated outside of the perspective of an agent. This hints at a choice as to the desired formalism's design: unlike a formalism such as Eigentrust, where components of trust scores are calculated independently of any one agent's perspective, the required responsibility scoring system must utilise a subjective perspective for *all* calculations. Strawson's argument applies to sociotechnical systems generally, as it does not apply to any specific trait or type of person; therefore, both social and technological actors in a sociotechnical system might be the targets of a trait designed with Strawson's work in mind.

Strawson also produces a fairly rigorous analysis of how ordinarily variable human traits can be formalised:

> Indignation, disapprobation, like resentment, tend to inhibit or at least to limit our goodwill

> towards the object of these attitudes, tend to promote an at least partial and temporary withdrawal of goodwill; they do so in proportion as they are strong; and their strength is in general proportioned to what is felt to be the magnitude of the injury and to the degree to which the agents will be identified with, or indifferent to, it.[18]

Strawson here identifies a similar property of human traits to Marsh: when an agent acts, the magnitude of the judgement of a trait with respect to the act is proportional to the magnitude of its importance. That is to say, if an action has a given importance, the magnitude of that importance should carry through to calculations of the judgements that the action impacts. This, combined with the subjectivity implied by the earlier point made by Strawson, implies that importance might be interpreted subjectively.

The introduction of a parameter of importance associated with a responsibility score allows the magnitude of importance of an act to be quantified and tracked in consequential or substantive responsibilities, adhering to Strawson's second point. Should some mechanism for interpretation be provided by the formalism, a subjective factor is introduced, satisfying Strawson's first point. This also hints at how an agent might judge the degree of responsibleness of itself or another agent, helping to answer the second research question; Strawson's argument here is therefore useful in the construction of the desired computational responsibility formalism.

3. A FORMALISM OF RESPONSIBILITY

A formalism is presented which, in theory, fulfils both research questions posed earlier in this paper. This formalism also adheres to the design principles laid out after analysis of the background research, as described in section 3.1.

3.1 Design Decisions and Philosophy

As was discussed through the background survey, many different philosophical decisions are made explicitly or implicitly during the creation of a formalism of an anthropic trait. In order to be certain of the finer details of the model created, then, it is important to clarify some of the choices made.

3.1.1 Social Accuracy

Firstly, the formalism created should be a Type-2 formalism by the distinction in section 1.2. The intention of the formalism is to create a proof-of-concept formalism, for which a general purpose, socially accurate formalism similar to Marsh's or Castelfranchi & Falcone's is ideal. A formalism which used responsibility as a metaphor would be inappropriate, as no purpose for the metaphor readily presents itself. In addition, no specific issues with existing formalisms are intended to be solved by the formalism — this is at odds with the background of a formalism such as FIRE, which was created specifically to solve the issues with existing anthropomorphic algorithms for trust. As current efforts in responsibility formalism have their roots in logic, no algorithmic formalism for responsibility exists to provide issues to solve. Therefore, the algorithm created should be a proof-of-concept with a focus on social accuracy.

3.1.2 Decentralised Information Exchange

The formalism created should also allow for decentralised calculations of responsibility importances and judgements of responsibleness. Centralised calculations of these properties might be useful for specific application areas — such as Eigentrust's focus on information security, where all agents communicate their states to all other agents so as to have all peers on the network as well-informed as possible. The desired responsibility formalism, however, should allow agents to operate individually, so as to create more socially accurate models (as not all information is guaranteed to be passed around a group of peers). One possible way to create a responsibility formalism with particularly well-informed peers would be to make use of recent work on computational gossip[2, 7]: this would allow peers to communicate responsibility metrics in a socially accurate way. However, this is beyond the scope of this proof-of-concept formalism.

3.1.3 Information the Formalism Assesses

The FIRE algorithm raises a second question as to the desired formalism's philosophy: FIRE is designed to solve the issue of judging information differently depending on its origin. A decision as to the information sources the designed formalism should consider should be considered. As the formalism created is intended to be a relatively uncomplicated proof-of-concept, the fewest information sources required to allocate, accept, discharge and judge responsibility should be modelled in this first formalism.

Work by both Sommerville and Scanlon indicate that a suitable degree of information for the judgement of responsibleness for causal responsibilities would be the previously discharged consequential responsibilities. Therefore, the formalism's responsibility judgement should focus primarily on the information recorded regarding these past responsibility discharges, their associated changes to sociotechnical state, and the successes of those state changes.

Strawson's writing provides some more useful insights regarding information needed when selecting responsibilities to discharge, as well as judgement of past responsibilities: each responsibility should have associated with it an importance, so as to consider Strawson's "magnitude of the injury" (or, here, sociotechnical state change). Therefore, information which this formalism should consider consists of consequential responsibilities of an agent, and those responsibilities' relative importances.

3.1.4 Responsibility Composition

Castelfranchi & Falcone's formalism of responsibility made clear that, in trust, the fundamental parameters they considered were a delegating agent and an agent subject to delegation, a desired action, and the goal the action ought to fulfil. Marsh, however, considered the action, but left his fundamental formalism devoid of a notion of goal.

In the desired responsibility formalism, nuances in the nature of a responsibility must be considered. Particularly, sociotechnical literature on a responsibility points to considering responsibilities as descriptions of state change, which ideally are composable. This state change could be considered the intended or required outcome of successfully discharging the responsibility — however, it does not imply any information regarding the action which brings about this state change.

Therefore, a responsibility formalism's fundamental pa-

rameters would appear to be an authority agent and a delegee, subject to responsibility delegation from the authority agent, plus a composable model of state change describing the responsibility itself. In addition, the desired formalism for this paper should include importance information as a fourth parameter, allowing for responsibility selection and responsibleness judgement as previously discussed.

3.1.5 Subjectivity

The required formalism ought to allow for subjective, agent-based "opinions" regarding the importances of modelled responsibilities. As importance is to be directly measured in this formalism, a mechanism by which an agent might interpret the importance scores of responsibilities would allow for a subjective measurement of responsibility importance, allowing an agent to perceive its own model of the trait according to its "opinions" about responsibility. These might be considered its preferences for responsibility selection, which weight the importances in its accepted responsibilities. A specific example of an interpretation function is given in section 3.3.6.

3.1.6 Graded Responsibility Calculation

A degree of responsibility should be calculable in a *graded* form, rather than binary. In Trust models such as Eigentrust, or Marsh's formalism, degrees of trust are represented numerically so as to better delineate between different trust measurements. By contrast, other models — particularly logical trust models — represent degrees of trust in a binary format, such as Castelfranchi & Falcone's basic form does.

The feature of graded measurement is useful for a few reasons. The feature is principally useful for comparison between many different agents when judging lots of responsibilities — task delegation amongst a pool of potential candidates being one scenario where this is useful. The feature therefore makes the formalism more pragmatic. Another benefit of a graded formalism is that it allows for a higher degree of social accuracy: human judgement of responsibility is much more nuanced than a simple boolean judgement.

Finally, a graded formalism allows for a deeper exploration of its efficacy. In order to evaluate the formalism properly, comparisons between agents with a responsibility formalism directing their behaviour and others with no such direction should be modelled.

3.2 Specific Formalism Details

Related research has helped to inform many design choices surrounding the formalism required to fulfil the research questions which this paper answers. As the necessary design decisions are now properly understood, a computational responsibility model which conforms to these design choices can be constructed.

3.3 Formalism Overview

Constraints, grouped into an obligation, are delegated by an authority to a delegee. The act of delegation implies enough information to permit the allocation of importance scores, completing the requirements for the creation of a responsibility. These importance scores are then interpreted subjectively by the delegee, who uses their present responsibilities to select future acts. Upon act completion, the associated responsibility is copied into a list of consequential responsibilities, and — should it not be listed as in the agent's notions — discarded from future discharge. These consequential responsibilities are then used to calculate basic, general, and specific responsibility scores, mirroring Marsh's seminal work on computational trust. Task selection, and alteration to interpretation coefficient via an advice mechanism, together answer the two research questions laid out during the problem specification.

The components of the formalism defined by this paper are constraints, obligations, responsibilities, importances, agents, actions agents can take to discharge responsibilities, an interpretation function, and a mechanism for judging the responsibleness of any given agent, and another for giving agents advice to improve their performance.

3.3.1 Constraints

A responsibility, for the purposes of this model, will be considered a state change in a environment. This definition is insufficient for *modelling* a responsibility, however, as metadata such as delegee, authority, and importance should also be defined; these are introduced to the constraint later in the formalism. In addition, it should be possible to compose these sociotechnical state changes.

To achieve this goal, constraints are introduced. Constraints are descriptions of a domain state change. Lock & Storer's idea of a composable responsibility form implies an atomic responsibility which cannot be *decomposed*; constraints satisfy this atomic property. Responsibility composition is described through the introduction of obligations and responsibilities in section 3.3.2 and section 3.3.3, however, composed constraints which have directly opposite effects will cancel out in practice. Composition of constraints is performed in such a way that the underlying responsibilities which cancel out are preserved, however, incase later analysis is required.

The model also defines two types of constraints:

ResourceDelta: a constraint which describes a set of changes to a number of variables (referred to here as a "resource") in the sociotechnical environment. The delta associated with the variable is described as a floating point number.

Deadline: the expected duration of the task. This constraint is special, because it is also provided with a clock for reference so as to calculate time relative to the constraint's reference frame. They are otherwise similar to a ResourceDelta.

Using these two constraint types, changes to the environment are recorded as specifications for a responsibility discharge. It is imperative to note that the constraint creation does not include an importance, even though the design philosophy of the formalism is to give each responsibility an importance. A constraint is the atomic definition of a responsibility's effect; therefore, all constraints must eventually receive importance scores. An importance score cannot be provided upon the creation of a constraint, however, as the constraint simply describes a state change — an importance score is a detail which only becomes necessary upon responsibility delegation, as different delegees may be required to meet the same state change specification with different importances. Constraints are therefore initialised without importance scores.

3.3.2 Obligations

Obligations are sets of constraints — they therefore allow constraints to be composed together. Obligations may only concern themselves with a single constraint, but defining a specific type which contains all constraints a responsibility is concerned with allows for a complete specification of that responsibility's effect on the environment.

An obligation therefore defines the complete domain effect which is defined by Lock & Storer as a responsibility — however, it concerns itself with no metadata as to the specific responsibility it serves as a specification of. Obligations might rather be considered a responsibility which has not been delegated: details such as authority and delegee are not specified, but the sociotechnical definition of the responsibility is fulfilled.

3.3.3 Responsibilities

The act of delegating necessarily defines certain details pertinent to a complete responsibility. These details include the authority and delegee associated with the responsibility — this relational information is defined by the act of delegation, as indicated by Strawson's relational take on anthropic traits explored in **??**

Importance scores can also be introduced at this point, as the delegee is defined; therefore, importance scores can also be assigned, as the only factor preventing importance score allocation upon the initialisation of a constraint is that the discharging agent — the delegee — was not defined, meaning that the importance of this specific agent discharging a responsibility was indeterminable. The importance scores are provided as floating points in the range $[0, 1]$, and are provided as a list in the same order as the constraints provided to the obligation.

A responsibility is used by an agent in the sociotechnical system for selecting a next task — to ease this calculation, the cumulative effect of a responsibility's discharge is defined as its "responsibility effect". The total effect of all constraints in its associated obligation, which maps factors affected to their cumulative effect. Defining this as a separate mapping allows for analysis of the original constraints which might be cancelled out upon composition in an obligation, and allows calculations to ignore or focus on any range of factors a responsibility concerns.

3.3.4 Agents

An agent consists of two important pieces of functionality:

- Interaction with a sociotechnical modelling framework.
- Task selection and responsibleness judgement using the responsibility formalism.

Agents also contain their own sociotechnical state. This should contain all of the variables that a obligation describes as being altered in some way. In addition, they are given a "basic responsibility score" — this number, in the range $[0, 1]$, represents a "benefit of the doubt" responsibility judgement for basic responsibility calculations. It is used in judgement of responsibility, which is described below.

To achieve subjectivity, agents contain interpretation coefficients; details regarding the interpretation coefficients are given in the interpretation function (section 3.3.6).

Agents should be defined with a set of responsibilities upon initialisation, which are referred to as notions. Notions represent responsibilities an agent may always discharge, such as an agent's option to idle away a clock tick without doing anything. Notions, unlike delegated responsibilities, are not removed from an agent's list of available responsibilities upon discharge. Notions are equivalent to the "norms" defined by Lock & Storer[16].

3.3.5 Acts

An act is supplied as a higher order function, paired with a responsibility effect, and is registered to an agent. This is done so that the expected effect of the function on the sociotechnical model can be compared against responsibilities that an agent has. When discharging a responsibility, all acts an agent has registered are compared by their effects against the expected effect of the responsibility discharge. If they match, the responsibility is discharged by executing the function associated with the act. Upon act completion, if the responsibility is not one of the agent's assigned notions, it should be removed from the list of available actions.

An act should set an "outcome" of each constraint of the responsibility an agent has currently selected, indicating whether the act successfully altered sociotechnical state during execution. This is used when calculating degrees of responsibleness of an agent.

After the act has successfully completed its execution, a copy of the discharged responsibility is saved into a list of consequential responsibilities, which is also stored within the agent. This allows later inspection of successful and unsuccessful constraint satisfaction for each consequential responsibility, which is used in judgement of an agent's degree of responsibleness.

3.3.6 Interpretation Function

Subjectivity within the formalism is achieved via an agent's interpretation function. The interpretation function makes use of an agent's interpretation coefficients: these are defined as a map of sociotechnical variables to floating point coefficients, which are used to weight the importance of each constraint in a newly delegated responsibility. Interpretation coefficients can be considered an agent's opinions or preferences regarding the importances of certain sociotechnical variables. Pseudocode for this interpretation procedure, as a method of an actor, is given in fig. 1.

```
function interpret(resp):
    for factor, coefficient in self.
        interpreting_coefficients:
        for constraint_index in range(length(resp.
            constraints)):
            constraint = resp.constraints[
                constraint_index]

            if factor in constraint.factors.keys:
                importance = constraint.importance
                    * coefficient
                constraint.importance = max(min(1,
                    importance), 0)  # Normalise
                !

            constraint.assign_importance(
                importance)
            resp.constraints[constraint_index] =
                constraint

    return resp
```

Figure 1: An example interpretation function in pseudocode

Some notable things present themselves in this block of code. For one, the method returns a new responsibility object. This means that a completely new responsibility, with altered importance information weighted by the interpreting agent's coefficients, is produced. This satisfies the subjectivity of an agent's interpretation. Old importance scores are retained as original importance scores of the constraint by the importance assignment method, however, meaning that a responsibility's original state can be restored for reinterpretation by another agent.

Another important note is the normalisation step after multiplying an importance by its coefficient. For some importances and coefficients, an agent may calculate an importance score outside of the defined range of $[0, 1]$. Therefore, a normalisation step is required to maintain the range.

All responsibilities processed by an agent are interpreted upon delegation; therefore, subjectivity of the formalism is maintained.

3.3.7 Directing Agent Decisions

As a result of responsibilities being interpreted after delegation, an agent can select a responsibility to discharge using the new interpretation scores. Mathematically speaking, responsibility selection is performed as follows:

$$new\_responsibility = argmax_{p \in resps} \sum_{i \in p.i} (i)$$

... if, for a responsibility denoted p, $p.i$ was the list of importances associated with the responsibility's constraints.

Using this, and accounting for all interpretation scores being interpreted already, *new_responsibility* becomes the responsibility with the highest cumulative importance. This responsibility is selected for discharge; the appropriate act to discharge the responsibility is found by a simple comparison between the responsibility's expected effect and the expected effect of each of the registered acts for the agent concerned. Example pseudocode to perform this lookup is given in fig. 2

```
function choose_responsibility():
    if self.responsibilities == []:
        return None
    else:
        resp = sorted(self.responsibilities,
                      key=lambda x: sum(x.
                          importances))[-1]
        return resp
```

Figure 2: An example responsibility selection using the formalism, in pseudocode

Note that, for ease of notation, Python's sorted function and list slices are used here, to indicate sorting by a given key and cleanly retrieving the last element in a list is used here. The simplicity of responsibility selection belies the simplicity of the formalism as a result of its strong data model. It is worth noting that, as responsibilities can be selected for discharge according to a subjective interpretation and discharged via the agent's registered acts, the formalism should answer research question 1 in theory. This is explored empirically in section 4.

3.3.8 Judging Agent Responsibility

An agent's degree of responsibleness can be calculated by a simple formula, which uses the outcomes of the consequential responsibilities of the agent to produce a basic, general, and specific responsibility score, in a fashion similar to Marsh's for trust[10]. An example general-purpose responsibleness degree function is given by the pseudocode in fig. 3.

```
function __judge_degree_responsible(subject agent)
    :
    specific_scores = dictionary()
    for each responsibility in subject agent's
        consequential responsibilities:
        reinterpret(responsibility)
        for each constraint in that responsibility
            :
            for each factor, delta in constraint:
                if factor not in specific_scores.
                    keys():
                    specific_scores[factor] = (0,
                        0)
                specific_scores[factor][0] += 1
                if constraint.outcome is True:
                    specific_scores[factor][1] +=
                        constraint.importance

    for each factor, score_tuple in
        specific_scores.items():
        count_for_factor = score_tuple[0]
        total_importances_discharged = score_tuple
            [1]
        specific_scores[factor] =
            total_importances_discharged /
            count_for_factor

    return specific_scores
```

Figure 3: Foundation of judgement of responsibleness, in pseudocode

This calculates the cumulative importance of all successfully discharged constraints of each factor, where a factor is a domain variable, altered by ResourceDelta constraints. These importances are re-weighted with the judging agent's interpretation coefficients in the "reinterpret" step. This is weighted against the number of all constraints of that factor — the result of which being that improperly discharged responsibilities do not count toward the degree of responsibility calculated. At the end of the algorithm, the dictionary specific_scores contains all *specific responsibility scores* calculable for the subject agent.

To calculate basic, general, and specific responsibleness judgements, where the above algorithm is implemented in the instance method `___judge_degree_responsible` via the specification in fig. 3, is given in fig. 4.

```
function basic_responsibility_judgement():
    return self.basic_judgement_responsible

function general_responsibility_judgement(
    other_agent):
    judgement = self.___judge_degree_responsible(
        other_agent)
    return mean(judgement.values())

function specific_responsibility_judgement(
    other_agent, resource_type):
    return self.___judge_degree_responsible(
        other_agent).get(resource_type, default=
        self.basic_judgement_responsible)
```

Figure 4: Granularities of responsibleness judgement, in pseudocode

A basic judgement of responsibility is here defined as the degree to which an agent believes other agents tend to be

responsible. It does not relate to any other agent specifically; therefore, the agent simply returns its basic judgement of responsibility.

A general judgement of responsibility is here defined as the degree to which another agent is responsible in all respects — it is therefore parameterised with the agent being judged. To calculate this, a mean is taken of all specific responsibilities which the agent calculates regarding the other agent.

Finally, a specific judgement of responsibility is here defined as the degree to which an agent is responsible for a particular thing, and is therefore parameterised by both the other agent involved and the factor the calculation relates to. Specific responsibilities are calculated by `___judge_degree_responsible` — therefore, a simple lookup of the dictionary returned by `___judge_degree_responsible` for the factor required is returned.

3.3.9 Updating the Agent Perspective Over Time

As an agent can judge any other agent's responsibilities by performing this algorithm upon the other's consequential responsibilities, an agent may perform this calculation on *itself*. Should an agent perform this calculation every time a responsibility is discharged, the agent may calculate the rate of change of its responsibility over time.

Improving its performance can be done in two ways. Should its interpreting coefficients ever change, the agent can reflect on its past performance and identify ways to improve; the agent would then readjust its interpreting coefficients to account for this introspection. While this behaviour is interesting, as it permits more socially accurate modelling, the agent's interpreting coefficients should be changed primarily through an *advice* mechanism.

An agent may advise another agent by offering that it take advice — this is represented as a dictionary of changes to the subject agent's interpreting coefficients. After judging the degree to which another agent is responsible, if an agent is seen to be lacking responsibility in a specific area, or in all areas generally (using the appropriate calculations), advice regarding the factors concerned is provided via this dictionary representation. The subject agent may choose to accept the advice depending on any implementation-appropriate factors: advising agent class, reputation formalism calculation, or other method.

This, combined with the judgement of responsibility provided in the earlier section, enables an agent to measure consequences of its actions via changes to its judgement of responsibility, and an analysis of its consequential responsibilities. The agent may also direct its interpretation of responsibility directly by offering itself advice, or taking the advice of others, changing the agent's interpreting coefficients — thereby theoretically fulfilling the second research question.

4. EVALUATING THE FORMALISM

At this point, the formalism's design is complete. Now, empirical data is required so as to verify that the formalism operates as intended, directing agent behaviour, and allowing for improved behaviour through introspection as to an agent's "beliefs" about the trait of responsibility.

The formalism is designed according to research-informed design philosophies, which should help to ensure that it answers the research questions positively. Investigating these research questions empirically requires producing data on responsible behaviour — however, quantifying responsible behaviour isn't something which is explicitly defined. The definition particularly depends on what type of responsibility one refers to: in this paper, for example, three ways to quantify responsibility have been determined, in the three judgements of responsibility outlined earlier.

In order to explore the efficacy of this formalism, one ought to take measurements using these judgements of responsibility. Simulations of sociotechnical systems must then be constructed so as to demonstrate responsible behaviour; these models ought not to have additional features or complications which could interfere with the data produced on responsible behaviour.

4.1 Experimental Design

In evaluating this formalism, a sociotechnical system was constructed to simulate the behaviour of students discharging responsibilities of coursework. A central authority, a Lecturer, was constructed to delegate tasks to students, and to judge all students equally (by applying their metrics for responsibility to all students). Importantly, Lecturers have no interpreting coefficients in this model: their judgement of responsibility is therefore untainted by the subjective nature of responsibility judgement, as re-interpretation by a Lecturer agent effectively resets a responsibility's importance scores to its original value. The result of this is that there is no inherent bias in the judgement toward any type of student.

Three types of agents are constructed to explore different aspects of responsibility.

StudiousAgents have interpreting coefficients which favour the resource `essay_writing` and `working_programs`.

HedonisticAgents have interpreting coefficients which favour the resource `personal_enjoyment`.

ResponsibilityBlindAgents have no responsibility formalism implemented: rather than choosing responsibilities which maximise importance, they choose a responsibility to discharge at random. This would be an example of a sociotechnical agent with no computational responsibility formalism to draw on.

The null hypothesis can be tested by comparison of either responsible agent class to the ResponsibilityBlindAgents: should there be no correlation between the formalism and agent behaviour, these agents should all perform identically with an agent with no formalism of responsibility whatsoever.

Importantly, all agents in this experiment are initialised with one Notion: the notion of Idling. This has the constraints of a Deadline of 1 tick, and a small amount of personal enjoyment. Its importances are given as 0.25 for all constraints. This notion is an important one, for two reasons:

- It effectively allows an agent to "waste" a turn, and relax a little (represented by the personal enjoyment constraint)
- As a result of the `personal_enjoyment` responsibility, HedonisticAgents will favour idling more than StudiousAgents will.

Agents are then given thirty responsibilities, evenly distributed among three types:

- Writing essays, constructed with a Deadline of 10 ticks and a ResourceDelta of `essays_written`: 1
- Writing programs, constructed with a Deadline of 10 ticks and a ResourceDelta of `working_programs`: 1
- Attending concerts, constructed with a Deadline of 10 ticks and a ResourceDelta of `personal_enjoyment`: 1

For each class of responsibility, each constraint was created with an importance of 0.075 * the number of constraints created. Therefore, every constraint created had an importance between 0.075 and 0.75, in steps of 0.075.

Agents were registered with acts which were capable of discharging these responsibilities, so as to conform with the formalism. In this way, agents were able to select from their pool of available responsibilities a single responsibility to discharge, and also select an act capable of discharging that responsibility according to the formalism's structure.

4.1.1 *Modelling Framework*

The entire simulation was implemented in Theatre_AG[15], a sociotechnical modelling framework with a focus on workflow modelling and monitoring reference frames across actors with respect to time.

Theatre was a suitable basis for the model for a number of reasons. Firstly, Theatre has a strong temporal model: its focus on consistent reference frames across actors made models with many agents feasible to implement easily. Theatre also presented a modelling platform with an emphasis on agent behaviour, as it was designed for modelling agent workflows. This initially presented a challenge, as Theatre did not originally support executing workflows which were constructed during the model's execution. Some of the time spent on research involved adding this feature, as well as Python 3 support, to the Theatre platform.

Theatre's `SynchronisedClock` object presented an ideal clock to feed into a Deadline constraint.

4.1.2 *Anticipated Outcomes*

The experiment was designed this way so as to show that a HedonisticAgent's behaviour would differ from that of a StudiousAgent. Should the agents perform similarly, this would imply that the responsibility formalism would not properly direct agent behaviour; thereby testing the first research question. Should the agents' behaviours differ, the general degree responsible (judged by the lecturer) would lessen for HedonisticAgents as the agents began to Idle, leaving many responsibilities undischarged. Meanwhile, StudiousAgents would continue to write programs and essays, increasing their general responsibility measurement; the two measurements would therefore deviate as the simulation progressed.

The second research question can then be tested, assuming the first research question is answered positively, by giving advice to the HedonisticAgents which altered their interpreting coefficients to more closely resemble those of the StudiousAgents. As HedonisticAgents would eventually stop writing essays, taking advice past this point would cause their behaviour to more closely resemble StudiousAgents; however, there would be a delay in the increase of the specific judgement of responsibility with respect to `essays_written` constraints, as the measure would increase only after advice was taken.

This should also be evident in measurements of general responsibility in advised agents, compared to unadvised HedonisticAgents or StudiousAgents, as an advised HedonisticAgent should cease to constantly Idle and would instead begin discharging `essays_written` and `working_programs` constraints.

4.2 RQ 1: Acting on Responsibilities

To answer the first research question, the experiment was conducted as described over the course of 150 simulated ticks in a Theatre_AG model. The graph produced is given in fig. 5.

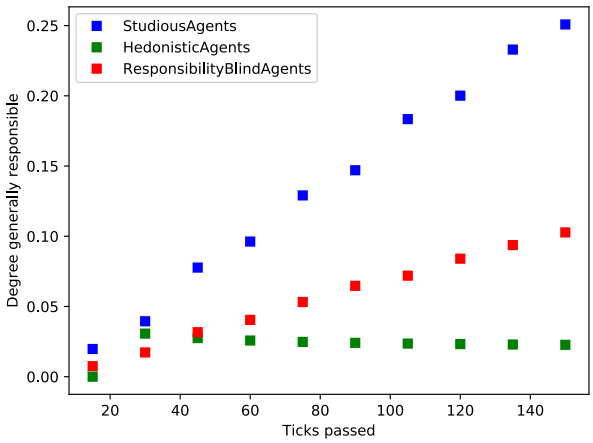

Figure 5: Agent behaviour over 150 ticks

Responsible agents' behaviours deviate from the ResponsibilityBlindAgents', successfully refuting the null hypothesis. Therefore, the responsibility formalism as outlined *does* direct agent behaviour. The agent behaviour is also directed subjectively, as can be seen in the disparity between StudiousAgents and HedonisticAgents.

Notably, while StudiousAgents perform better than ResponsibilityBlindAgents in terms of general measure of responsibility, HedonisticAgents do not. This is an important result: using a responsibility formalism, an agent can perform better than random task selection, making a responsibility a viable task scheduling mechanism which deviates from traditional methods. If improperly calibrated, however, the formalism can perform *worse* than random task selection. This shows that an agent's ability to improve on its performance by directing it's interpretation of responsibility is crucial to real-world applications of the formalism.

The predicted plateau of the HedonisticAgents' behaviour is also seen after roughly 30 ticks elapse. This indicates that the interpreting coefficients caused these agents to idle, no longer completing additional tasks, as expected.

Research question 1 is answered positively, as a result of the divergent behaviour of all three agent types. Research question 2 should now by explored, as agent behaviour is affirmatively directed by the formalism, but improvements in performance as interpreting coefficients change is yet to be seen.

4.3 RQ 2: Judgement of Responsibility

A requirement of exploring the second research question is that the advice mechanism outlined in the formalism's design should be utilised. An experiment was therefore devised where HedonisticAgents were advised by a lecturer to adjust interpreting coefficients to resemble those of a StudiousAgent. Advice was given after 100 ticks had elapsed. The resulting simulation can be seen as graphed in fig. 6 with specific responsibility judgement for all agents regarding the resource `essays_written`, and in fig. 7 with general responsibility judgement.

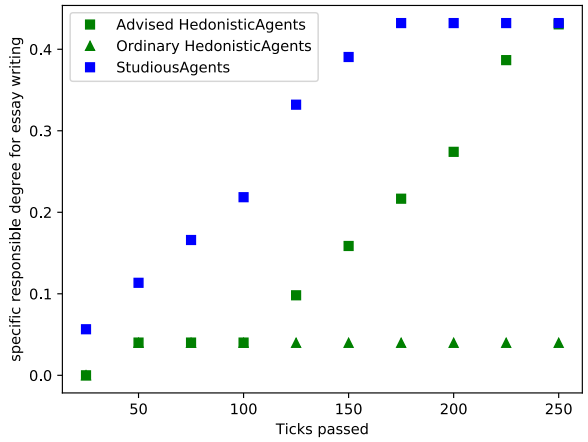

Figure 6: Specific judgement of `essays_written` responsibility comparing HedonisticAgents with and without advice

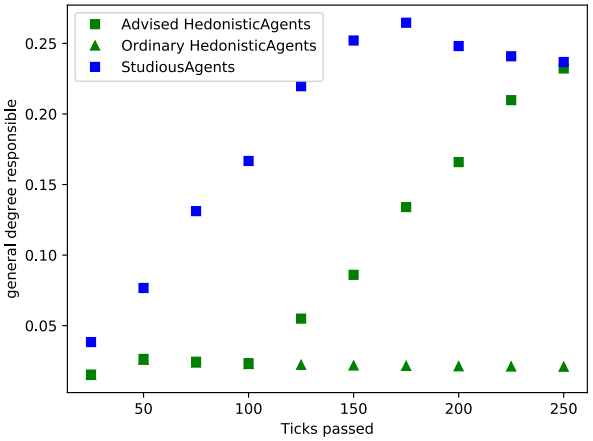

Figure 7: General judgement of responsibility comparing HedonisticAgents with and without advice

As can be seen, immediately after accepting advice, HedonisticAgents deviated from their ordinary behaviour and behaved as StudiousAgents had done roughly 100 ticks prior. This pattern is seen in both fig. 6 and fig. 7, meaning that the advice successfully affected both general and specific responsibility measurements. The outcome of the experiment therefore answers the second research question affirmatively: via the advice functionality, a responsible agent's behaviour can be updated.

It is important to note, however, that agents were not providing their own advice. Agents were being identified by a Lecturer agent as performing poorly (as they were HedonisticAgents), asserting that their performance was flawed in some way, and then providing a solution to this flaw in the form of advice. Two problems must yet be solved for agents to supply their own advice:

1. Agents must be able to identify their own poor performance.
   As poor performance is identified with interpreted importance scores, an agent will always "believe" that it selected optimal responsibilities for discharge. Soliciting advice from other respected agents, or reinterpreting importance scores with the coefficients of other respected agents, is necessary for the proper assessment of poor performance.

2. Agents must be able to identify corrective measures themselves.
   Identifying improper behaviour is a separate task to identifying how to make that behaviour better. Unfortunately, the numbers produced by responsibility judgement do not provide adequate information for identifying corrective behaviour; therefore, additional mechanisms for judgement must be constructed.

The second research question was therefore answered affirmatively, but solving these two additional problems is required for agents to correct their own behaviour independently. While the first problem can be solved with relatively trivial alterations to the presented formalism, the second problem stands as a new research question in its own right.

5. FUTURE WORK

5.1 Independent Behaviour Direction

The two problems outlined as a requirement to full behaviour direction at the end of section 4.3 must be solved for the complete independent direction of an agent's behaviour. The first outlined problem is technically solved with a minor modification to the existing formalism; however, the second stands as its own separate research question.

In order to solve the second question, an agent may be required to adopt subsets of another agent's interpreting coefficients, or to judge responsibleness in a more nuanced and informative way. One way to assess alternative decisions, and to update interpreting coefficients, would be to build an alternative stack trace of choices another agent would have made, given the same consequential responsibilities. Two potential issues arise with this method, however:

1. If the other agent had accepted advice in the past, this would need to be taken into account when building a stack trace. Currently, no history of advice taken is kept, making the stack of alternative decisions impossible to properly produce.

2. No history is kept of the other responsibilities available when each choice to discharge a responsibility is made. This history would be vital to keep, in case delegation revoking were included in a later version of the formalism, or more in-depth analysis on the choice made was required.

5.2 Repeated Acts to Discharge a Responsibility

Currently, an action cannot be repeated multiple times to discharge a responsibility; instead, a responsibility is discharged only if an act is available with a perfectly matching expected effect.

Similarly, some acts may discharge a responsibility with additional side-effects. These do technically fulfil a responsibility's requirements, and should also be included in an ideal formalism.

5.3 Applying The Formalism

The formalism has been shown to have potential in task scheduling applications. Further work on the formalism, developing enhanced versions with design philosophies more appropriate for specific applications, would prove the real-world and industrial applicability of this research.

The formalism at present may already be suitable for real-world application for task scheduling. Empirical evidence of this, and any modifications to improve the formalism's performance in more constrained scenarios, will require further research, posing new research questions.

5.4 Integration With Trust

Similarities between trust and responsibility as social traits were laid out in earlier sections of this paper. A wealth of insights from trust formalism, including the basic parameters required to construct the formalism presented, were incorporated into the final product.

Due to these similarities, and the employment of similar design philosophies between the formalism presented and existing trust formalisms, such as Castelfranchi & Falcone and Marsh's respective formalisms, development of a model which incorporates the two traits appears feasible. Each trait's measurements might be used to inform the other in a multi-trait anthropomorphic algorithm, creating a feedback mechanism which allowed each trait to be calculated by a better-informed agent. Multi-trait formalisms usually exist in a format similar to Eigentrust's employment of reputation as a bootstrapping trait for trust; this proposed multi-trait model would instead present two separate traits, computed in parallel, but informing each other.

5.5 Focused Anthropomorphic Algorithms Research

5.5.1 Motivation for a New Field

The term "anthropomorphic algorithm" is introduced in this paper as a way of separating computational formalisms of human behaviour from other research in sociotechnical systems and human-like computing. The term is introduced here because this separation of concerns matters: a research focus on these computational formalisms allows for study of the behaviour being formalised.

Current research methods often attempt to study these behaviours as emergent phenomena, but an anthropomorphic algorithms focus creates a level of abstraction between the behaviour modelled and the underlying social and technical factors which create them in the real world. This abstraction is useful: modelling the fundamental cognitive processes which lead to social behaviours in humans is a notoriously intractable problem in psychophysics. To formalise these fundamental traits, create an understanding of their complex neural origins, and construct simulations of this behaviour for other research purposes presents a mammoth interdisciplinary research effort, which would be impossible with current state-of-the-art research in both social sciences and computer science.

A more appropriate angle of approach would be to model the trait directly, arriving at the same phenomenon which grows naturally from more complicated principles than those formalised. In this way, the *behaviour* can be studied without the need to solve currently intractable problems across several disciplines, and without the need to communicate highly complex ideas across disciplines. Indeed, this approach is taken implicitly in current research in trust, comfort and other traits. An explicit research focus in anthropomorphic algorithms would create a community of researchers across disciplines working toward the common goal of trait formalisation.

5.5.2 Anthropomorphic Algorithms-Focused Modelling

Not only would this interdisciplinary knowledge gap become easier to overcome (due to the additional degree of abstraction), but more sophisticated modelling techniques can be composed to cater to the specific needs of anthropomorphic algorithms. In particular, no modelling platform for research on and evaluation of anthropomorphic algorithms currently exists. Evaluation of the responsibility formalism presented relied on the Theatre_AG library for its temporal model; a suitable anthropomorphic algorithm model might also contain components such as:

- A sociotechnical environment model, so as to allow agents to interact with a well-specified environment
- An agent model, allowing for different agent modelling techniques. For example, within the field of responsibility, the formalism presented models agents as individuals who act independently; other traditional modelling technques, such as Sommerville et al.'s methods, model the roles that arbitrary actors might inhabit.
- A tempoal model, such as a ticking model as provided by Theatre_AG or popular sociotechnical models such as timebands[3]. Other temporal models are also popular within the formal methods community.
- Agent behaviour modelling — agent behaviours within anthropomorphic algorithms are provided by the formalisms the algorithms implement; multi-trait formalisms currently tightly couple different traits together, such as Eigentrust's melding of trust and reputation. Multi-trait formalisms such as those described in section 5.4 couple be easier to research with behaviour modelling which enables formalism design with looser trait coupling.

Standardising these factors, as well as others, which are necessary for a complete specification of an anthropomorphic algorithm would allow a modelling framework to be constructed specific to the field. This would provide a common jargon across experts in social sciences, mathematics, and computer science, easing interdisciplinary research. It would also encourage improved engineering when designing multi-trait formalisms, producing looser-coupled multi-trait models, as well as enabling consistent evaluation metrics and modelling techniques across the research community.

Defining these necessary components and exploring possible modelling platforms for anthropomorphic algorithms is therefore an important first step in cementing anthropomorphic algorithms as an independent area of research.

6. DISCUSSION

In this paper, a new formalism of computational responsibility was presented which is able to direct agent behaviour in a subjective way toward discharging the most important responsibilities earliest.

This formalism employs a wealth of insights from sociotechnical literature, as well as literature on trust modelling, the trait with the largest body of anthropomorphic algorithm development to date. In this paper, the distinction of anthropomorphic algorithms from the rest of sociotechnical literature is defined, as well as its distinction from human-like computing.

The presented formalism, applied to a simple sociotechnical modelling scenario, effectively shows that responsibility as it is presented can direct agent behaviour, and is sufficient for significant behavioural changes. It is also shown to perform better than random task selection, where responsible behaviour is required. Improvements to agent behaviour are also demonstrated, ensuring that under-performing agents can become more performant over time by employing the formalism's advice functionality.

One novel contribution of the work was the introduction of importance scores, and their subjective interpretation. Using this method, divergent behaviour of agents was achieved from the same initial conditions. The definition of responsibility as a sociotechnical state change, delegated from one agent to another with a specific importance for each individual change it describes, was also an important contribution — similarities between responsibility and trust are evident in this definition, while responsibility's nuances are also respected. The definition kept to its design philosophies while permitting functionality seen in other formalisms, such as Marsh's three granularities of judgement and the simplicity of formalisms such as Castelfranchi & Falcone.

The formalism successfully answers the research questions — and answers both affirmatively — however, it demonstrates the need for further research focus on the trait of responsibility. Presented is a proof-of-concept that responsibility is a valuable trait in anthropomorphic algorithms research; its ultimate application and broader utility can be uncovered only by further work.

7. REFERENCES

[1] F. Berreby, G. Bourgne, and J.-G. Ganascia. Modelling moral reasoning and ethical responsibility with logic programming. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 532–548. Springer, 2015.

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14(SI):2508–2530, June 2006.

[3] A. Burns and G. Baxter. Time Bands in Systems Structure. *Structure for Dependability: Computer-Based Systems from an Interdiciplinary Perspective*, pages 74–88, 2006.

[4] C. Castelfranchi and R. Falcone. Social Trust: A Cognitive Approach. 2001.

[5] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 23–27. IOS Press, 2004.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.

[7] J. Lavaei and R. M. Murray. Quantized consensus by means of gossip algorithm. *IEEE Transactions on Automatic Control*, 57(1):19–32, Jan 2012.

[8] R. Lock, T. Storer, I. Sommerville, and G. Baxter. Responsibility modelling for risk analysis. 2010.

[9] S. Marsh, P. Briggs, K. El-Khatib, B. Esfandiari, and J. A. Stewart. Defining and investigating device comfort. *Information and Media Technologies*, 6(3):914–935, 2011.

[10] S. P. Marsh. Formalising Trust as a Computational Concept. *Computing*, Doctor of(April):184, 1994.

[11] T. M. Scanlon. Justice, responsibility, and the demands of equality. 2006.

[12] R. Simpson and T. Storer. Socio-Technical Modelling Techniques : A Case-Study Comparison. 2011.

[13] I. Sommerville. Causal responsibility models. In *Responsibility and Dependable Systems*, pages 187–207. Springer, 2007.

[14] I. Sommerville. Models for responsibility assignment. In *Responsibility and dependable systems*, pages 165–186. Springer, 2007.

[15] T. Storer. Theatre_ag. `https://github.com/twsswt/theatre_ag`, 2017.

[16] T. Storer and R. Lock. Modelling responsibility. Technical report, Project Working Paper 7, InDeED Project (April 2008), 2008.

[17] T. Storer, S. Marsh, S. Noël, B. Esfandiari, K. El-Khatib, P. Briggs, K. Renaud, and M. V. Bicakci. Encouraging second thoughts: Obstructive user interfaces for raising security awareness. In *PST*, pages 366–368, 2013.

[18] P. F. Strawson. Freedom and resentment. *Proceedings of the British Academy*, 48:1–25, 1962.

[19] P. Wallis and F. Cloughley. *The OBASHI Methodology*, volume 1. The Stationery Office, 1 edition, 2010. The offical manual for the OBASHI Methodology.

[20] T. Wallis. Anthropomorphic algorithms [https://youtu.be/RGUeYQzRsOQ]. Let's Talk About [X], 2017.