University of Glasgow | School of Computing Science

Investigating Computational Responsibility

William Wallis (2025138)

April 10, 2017

ABSTRACT

Currently, models are produced for responsibility modelling which have their roots in logic. These models, while sophisticated, suffer from a lack of pragmatism: for guiding agent behaviour in sociotechnical simulations, logical models are not always ideal. In the similar field of trust modelling, algorithmic models which emulate social behaviour produce useful results while being easier to understand, implement, and reason about. In this paper, a proof-of-concept responsibility modelling platform adopting the algorithmic formalism style employed by trust modelling is produced, and its utility evaluated.

1. INTRODUCTION

A growing area of research lies in the formalism of human traits into computational representations. These algorithms make computers more human-like; for that reason, they are referred to here as "Anthropomorphic Algorithms". 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 report.

This implementation interest, realised in the study of anthropomorphic algorithms, presents an interesting sociotechnical problem. They present an opportunity to alter the behaviour of actors in a sociotechnical system, and to do so in a way that is easy to reason about. This alternation of behaviour is done by the algorithmic implementation of a *formalism*. Formalisms present a concrete definition — by process, mathematical definition or semantic description — which can be used to construct an anthropomorphic algorithm. These formalisms tend to attempt to model in one of two ways:

1. Modelling the 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.

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.

In reality, most formalisms and their implementations lie somewhere on the spectrum that these two types define.

2. STATEMENT OF PROBLEM

Computational formalisms of human traits are a growing field of research, with applications in lots of different areas. A problem with these anthropomorphic algorithms is that there is limited breadth to the scope of existing research in the field (as is demonstrated during the background survey in section 3). The metaphor of the human trait in these algorithms remains largely unexplored.

Breadth in the application of the metaphor is important, however. The importance stems from the utility in the human metaphor when designing systems:

- Human-Computer Interaction can make use of behavioural metaphors to relay complicated internal states to a user. Storer et al. [18] demonstrated methods by which a mobile device might dissuade certain user actions by expressing its "discomfort" or lack of "trust" in its interaction design.

- Information Security can make use of behavioural metaphors in order to increase difficulty of access when negative system states are encountered. A system might allow access on a graded scale, dependant on internal states of trust, comfort, and confidence.

- Theoretical advancements in smart city technology[21] might increase a city's resilience by integrating notions of responsibility into public services and the environment on a community scale.

While similar results can often be achieved using regular techniques, the human metaphor allows for a better communication between a human user and complicated system states. All of the above examples center around this notion; however, the applications extend beyond Human-Computer Interaction research.

The lack of application of the human metaphor is a complicated mosaic of related factors. For example, research into anthropomorphic algorithms holds particular challenges, as a result of its strongly interdisciplinary nature: it requires a research team to understand the nuances of sciences as well as social sciences, and sometimes even humanities. Not only does the research team require the ability to understand these nuances, but the research must take into account their different natures. This often causes divergence in the philosophies of sociotechnical research. Some researchers view sociotechnical systems from the perspective of largely human-based systems with abstract, social behaviour. Others see sociotechnical systems as a combination of dynamic, mathematical processes which produce more technical emergent phenomena. This hints at a third complicating factor, (the second being a lack of convergence in research focus): a lack of a consistent modelling paradigm. Some research focuses largely on actor interaction-style modelling techniques [1], while others rely on purely graphical modelling [11], or on mathematical modelling techniques [19].

These issues together pose an issue for research in anthropomorphic algorithms: a formalism of a human-like trait is only useful to certain researchers, for certain types of models, with certain sociotechnical philosophies. Their lack of broad application is therefore unsurprising; these factors compound to produce yet another, which is that the breadth 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 recently been attempted also [9].

Recently, some interest has been shown in research pertaining to modelling and formalising *responsibility.* Logical models of social trust exist which could be turned into a proof-of-concept formalism of responsibility with only a small addition: adding an obliging term in a similar way to the Deontic Logic's system for obliging[20], allowing an agent to effectively delegate a task to another which is deemed responsible in discharging responsibilities — the agent selected being known to be trusted already, by C&F's already established work. The scheduling of tasks based on trust is a simple extension of existing models, or an application of existing models.

A model of responsibility might be more than simple task allocation, however. Some logical models of responsibility attempt to model ethically responsible decisions [2]. Deontic logic's obliging term was in itself an attempt to create a logic which was suitable for the calculation of whether an agent was obliged to ensure certain goals, or perform certain tasks. Another angle might be to perceive a model of responsibility as something which might allow a responsible sociotechnical agent to choose responsibilities to discharge, rather than blindly executing tasks they are provided with in a trust-oriented model which simply delegates tasks.

The latter has a number of potential applications. One such possibility would be to implement agent awareness of remote task execution via RPC. Should an agent on a network be given a procedure to execute which is perceived through a responsibility formalism to be unusual, the procedure may not be executed, or may be rejected upon receipt by the responsible agent. Similar applications have proven effective in trust literature, particularly the Eigentrust algorithm [6] , where information security is enhanced by inferring agent trustworthiness.

In a realistic sociotechnical system, 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. In acknowledging that 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?

Answering these questions requires the construction of a proof-of-concept formalism of responsibility which is suitable for directing agent behaviour in an algorithmic manner, as opposed to existing logical methods.

3. RELATED WORK

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

3.1 Trust Modelling

Ordinarily, when constructing a new anthropomorphic algorithm, one would draw on literature regarding other anthropomorphic algorithms which formalise the trait being modelled. 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 (as investigated earlier).

3.1.1 Marsh

The seminal anthropomorphic algorithm for trust is found in Marsh's 1994 formalism[10]. In this paper, a formalism 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 granularity in trust judgement. Marsh identifies that human agents have a basic degree or weighting which applies to their trust — it would not be uncommon to assert that "person X is very *trusting*". This is, however, distinctly different to an assertion that "person X is trusting, but *doesn't trust person Y*" — that is, a person's basic level of trust is distinctly different from a person's more directed degree of trust toward another agent in particular. Marsh calls this "General Trust", differentiating it from the earlier "Basic Trust". A final distinction, "Specific Trust", 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).

These separations are useful in a few key ways. One is that it identifies some of the key 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". 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 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 more nuanced comparison between agents.

3.1.2 Castelfranchi & Falcone

Another useful formalism of trust to consider comes from Castelfranchi & Falcone [4] (often referred to as "C&F theory"). While this formalism has logical foundations, its large popularity makes it an interesting comparison to Marsh's formalism.

In contrast to Marsh's formalism, C&F is graded only in more complex forms. At its root, C&F is a logical formalism built on boolean calculations of goal and belief state satisfaction. However, C&F 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 C&F 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 it. Therefore, C&F uncovers the need when building a formalism of a trait to make choices as to the formalism's philosophy regarding the trait.

In particular, two philosophical differences between Marsh's model and that of C&F are immediately apparent: where Marsh separates his calculation of trust into a basic/general/specific granularity, C&F 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 report.

3.1.3 Eigentrust

Eigentrust[6] is a formalism which takes a notably different approach to Marsh and C&F'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 C&F's respective formalisms:

- Eigentrust operates in a distributed way. Unlike the local scoring system employed by Marsh and C&F's formalisms, Eigentrust has all agents report their local trust scores, so as to create a distributed ledger of more general trust scores.
- Eigentrust does not model trust directly, nor does it claim to model it accurately — unlike C&F's formalism, which is strictly intended as a model of human behaviour, and Marsh's, which simulates it, Eigentrust uses "trust" as a 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 C&F'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.

3.1.4 *FIRE*

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. 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 philosophical 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 philosophical 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.

3.2 Sociotechnical Systems

While these anthropomorphic algorithms 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.

3.2.1 *Ian Sommerville*

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 for information — 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[11].

Sommerville's writing, however, presents a wealth of interesting insights which may be useful in understanding the context of the 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 approach.

3.2.2 *Timothy Storer & Russell Lock*

While describing a graphical responsibility modelling format for the InDeED project — notably lead by Ian Sommerville — Storer and Lock provide some useful foundations for other models of responsibility in a technological report on modelling responsibility[16].

Most interestingly, Storer and Lock provide a useful definition 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 Mash and C&F, 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*. Storer and Lock 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 — Storer and Lock note that these responsibilities must not conflict with norms that an agent holds.

Unlike Marsh or C&F'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 would provide a very useful starting point for the desired responsibility formalism.

Another useful definition from Storer and Lock'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.

3.3 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 report 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.

3.3.1 Thomas Scanlon

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[12]

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 posesses 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 — though clarity on how this distinction is useful is found in work from another philosopher, P.F. Strawson.

3.3.2 P.F. Strawson

P.F. Strawson is a philosopher with work on philosophy of language and of mind — one particular essay, "On Freedom and Resentment"[17], presents an argument that determinism shouldn't affect how we percieve human factors such as responsibility or trust.

Strawson argues that these concepts are fundamentally relational, rather than having any inherant 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 philosophy: 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, 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.[17]

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 effect on an overall judgement of the trait. That is to say, if an action has a given importance, the magnitude of that importance should carry through to 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.

3.3.3 Sloman

4. A FORMALISM OF RESPONSIBILITY

4.1 Design Decisions and Philosophy

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

4.1.1 Social Accuracy

Firstly, the formalism created should be socially accurate. No specific application area presents itself for this formalism; rather, the work presented is intended as a proof-of-concept for anthropomorphic algorithms pertaining to responsibility. 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 computational responsibility have their roots in logic, no anthropomorphic algorithm for responsibility exists to provide issues to solve. Therefore, the algorithm created should be a proof-of-concept with a focus on social accuracy.

4.1.2 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[3, 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 project.

4.1.3 Information the Formalism Regards

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 responsibileness 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.

4.1.4 Responsibility Composition

The content of a responsibility can be somewhat clarified at this point. In particular, C&F'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 parameters 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 projectshould include importance information as a fourth parameter, allowing for responsibility selection and responsibleness judgement as previously discussed.

4.1.5 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 percieve 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 the specifics of the formalism designed below.

4.2 Specific Formalism Details

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

The components of the formalism defined by this paper are Constraints, Obligations, Responsibilities, Importances, Agents, an Interpretation Function, and a mechanism for judging the responsibleness of any given agent.

4.2.1 Constraints

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

To achieve this goal, Constraints are introduced. Constraints are descriptions of a sociotechnical state change. Storer & Lock's idea of a composable responsibility form implies an atomic responsibility which cannot be *decomposed* — Constraints satisfy this atomic property.

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 — clocks are provided by the Theatre modelling library[15]. They are otherwise similar to a ResourceDelta.

Using these two constraint types, changes to the sociotechnical 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 a formalism. 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 sociotechnical 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.

4.2.2 *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 sociotechnical environment.

An Obligation therefore defines the complete sociotechnical effect which is defined by Storer & Lock 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. Additional information is supplied only when this Obligation is delegated.

4.2.3 *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.

Importance scores can also be introduced at this point, as the delegee is defined; therefore, information scores can also be assigned, as the only factor preventing information score allocation upon the initialisation of a Constraint is that the discharging agent — the delegee — was not defined. 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 aid this, Responsibilities provide some convenience methods. In particular, Responsibilities allow for calculating specific details regarding the Obligation it is initialised with, such as the cumulative effect of the various Constraints it is defined by. In the implementation for testing this formalism, the effect of a Responsibility was contained in a ResponsibilityEffect object — this allowed for providing further convenience methods, allowing one to get the total effect on one particular resource (from all contained ResourceDelta Constraints), or to disregard any particular resources which were effected when reasoning about the effect of a Responsibility.

4.2.4 *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.

Useful sociotechnical modelling frameworks for utilising the formalism are workflow modelling frameworks which maintain a consistent reference frame across all actors. Should the framework allow for online workflow modelling — workflows where the next item in the workflow is not predetermined — then a workflow can be constructed during the model. This permits a non-deterministic model execution, which is vital, as task selection according to the formalism is subjective.

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 are expected to contain interpretation coefficients; details regarding the interpretation coefficients are given in the Interpretation Function section-section 4.2.6.

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 Storer & Lock[16].

Agents are complicated entities which implement the entire anthropomorphic algorithm described in this report— two particular elements of the Agent's implementation are Acts and the Interpretation Function, which are detailed below, in their own sections.

4.2.5 *Acts*

A workflow model is particularly useful for the construction of Acts.

An Act is supplied as a Higher Order function, paired with a ResponsibilityEffect, and are registered via a method on 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,

Using a workflow modelling framework such as Theatre[15], the functions associated with the act can be registered by using the functions contained within the workflow as the higher-order functions.

An act should set an "outcome" attribute 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 an attribute of the agent. This allows later inspection of successful and unsuccessful constraint satisfaction for each consequential responsibility, which is used in later judgement of an agent's degree of responsibleness.

4.2.6 *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. Example Python code for this interpretation procedure, as a method of an actor, would look as follows:

```
def interpret(self,
              resp: Responsibility):
    resp = copy(resp)
    for factor, coefficient in self.
         interpreting_coefficients.items():
        for constraint_index in range(len(resp.
             constraints)):
            old_constraint = resp.constraints[
                 constraint_index]
            constraint = copy(old_constraint)
            importance = constraint.importance

            # Work out the new importance value, if
                 there is one.
            if factor in constraint.factors.keys() or
                 factor == constraint.___class___:
                importance = importance * coefficient
                importance = max(min(1, importance),
                     0)  # Normalise!

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

    # Return the responsibility with a new set of
         constraints
    return resp
```

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 by the Constraint.assign_importance() method, however, meaning that a responsibility's original state can be restored for re-interpretation 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 set 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.

4.2.7 *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:

$$new\_responsibility = argmax_{p \in resps} \sum_{i=p.i[1]}^{p.i[length(p.i)]} (i)$$

If, for a Responsibility denoted p, $p.i$ was the list of importances associated with the responsibility, with indexing beginning at 1.

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 Python code to perform this lookup might be:

```
def choose_responsibility(self):
    if self.responsibilities == []:
        return None
    else:
        resp = sorted(self.responsibilities,
                      key=lambda x: sum(x.importances)
                         )[::-1][0]
        return resp
```

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. The evaluation section of this report, found below, confirms this.

4.2.8 *Judging agent responsibility*

An agent's degree of responsibleness can be calculated by a simple formula,rithm, 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]. The specific responsibility generation might be represented in the following way in pseudocode:

```
def __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
```

This calculates the cumulative importance of all successfully discharged constraints of each factor, where a factor is a sociotechnical variable. 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 specific, general, and basic responsibility scores, the following Python code might suffice, where the above algorithm is implemented in the instance method "___judge_degree_responsible":

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

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

def specific_responsibility_judgement(self,
    other_agent, resource_type):
    return self.___judge_degree_responsible(other_agent
        ).get(resource_type, self.
        basic_judgement_responsible)
```

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, which is a simple object property.

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. The

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.

4.2.9 Updating the sociotechnical opinion 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.

Each agent is given a method, "advise", and another, "take_advice". An agent may advise another agent by offering that it take advice — advice 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 in responsibility in a specific area, or in all areas generally (using the appropriate calculations), advice regarding concerning factors 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.

4.3 An Overview of the Formalism's Definition

A formalism has now been presented which, in theory, fulfils both research questions posed earlier in this report. This formalism also adheres to the design principles laid out after analysis of the background research.

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 interpretation scores, completing the requirements for the creation of a Responsibility. These interpretation 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 consideration. 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 together answer the two research questions laid out during the problem specification.

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

5. EVALUATING THE FORMALISM

5.1 RQ 1: Acting on Responsibilities

5.2 RQ 2: Judgement of Responsibility

6. FUTURE WORK

6.1 Repeated Acts to Discharge a Responsibility

6.2 Advancing the formalism

7. DISCUSSION

8. REFERENCES

[1] G. Baxter and I. Sommerville. Socio-technical systems: From design methods to systems engineering. *Interacting with computers*, 23(1):4–17, 2011.

[2] 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.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14(SI):2508–2530, June 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] F. C. Paul Wallis. *The OBASHI Methodology*, volume 1. The Stationery Office, 1 edition, 2010. The offical manual for the OBASHI Methodology.

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

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

[18] V. T. Tim Storer, Karen Renauld. Find this find this. 2020.

[19] A. Vespignani. Modelling dynamical processes in complex socio-technical systems. *Nature Physics*, 8(1):32–39, 2012.

[20] G. H. von Wright. Deontic logic. *Mind*, 60(237):1–15, 1951.

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