Abstract

Trust is a mechanism for managing the uncertainty about autonomous entities and the information they store, and so can play an important role in any decentralized system. As a result, trust has been widely studied in multiagent systems and related fields such as the semantic web. Here we introduce a formal system of argumentation that can be used to reason using information about trust. This system is described as a set of graphs, which makes it possible to combine our approach with conventional representations of trust between individuals where the relationships between individuals are given in the form of a graph. The resulting system can easily relate the grounds of an argument to the agent that supplied the information, and can be used as the basis to compute Dungian notions of acceptability that take trust into account. We explore some of the properties of these argumentation graphs, examine the computation of trust and belief in the graphs, and illustrate the capabilities of the system on an example from the trust literature.

1 Introduction

Trust is a mechanism for managing the uncertainty about autonomous entities and the information they deal with. As a result, trust can play an important role in any decentralized system. As computer systems have become increasingly distributed, and control in those systems has become more decentralized, trust has been an increasingly important concept in computer science [6, 35].

As a number of authors have pointed out, trust is a concept that is both complex and rather difficult to pin down precisely and as a result, there are a number of different definitions in the literature. Thus, to pick a few specific examples, Sztompka [85] suggests that:

Trust is a bet about the future contingent actions of others.

while Mcknight and Chervany [61], drawing on a range of existing definitions, define:

*This paper is a revised and expanded version of [86].
Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.

and Gambetta [28] states:

Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends.

In the multiagent systems community, an influential model of trust is that of Falcone and Castelfranchi [25] who argue that trust arises when the trusting agent only trusts another with respect to some goal, and thus with respect to some action that the second agent can perform which will bring about that goal. Trust then relates to the beliefs that the trusting agent holds, which include a belief that the second, trusted agent is capable of bringing about the goal, will carry out the action to bring about the goal, and that this action is necessary to bring about the goal.

It is also pointed out in a number of places that there are different kinds of trust, what [44] calls “trust scopes”. For example, [35] identifies the following types of trust:

1. **Provision trust:** the trust that exists between the user of a service or resource, and the provider of that resource.
2. **Access trust:** the trust that exists between the owner of a resource and those that are accessing those resources.
3. **Delegation trust:** the trust that exists between an individual who delegates responsibility for some action or decision and the individual to whom that action or decision is delegated.
4. **Identity trust:** trust that an individual is who they claim to be.
5. **Context trust:** trust that an individual has in the existence of sufficient infrastructure to support whatever activities that individual is engaged in.

While these definitions of trust vary, there are clearly some common elements. First, there is a degree of uncertainty associated with trust. As [11] point out, trust is a function of the subjective certainty of the trusting agent’s beliefs. This subjective certainty might, as argued by Gambetta, be expressed as a subjective probability, or as Sztompka prefers to think of it as a bet. The subjective certainty can be expressed as a “feeling of security” [61], or as Castelfranchi and Falcone [11] prefer, in terms of reasons based on perceived risk. Second, trust is tied up with the relationships between individuals. Trust is related to the actions of individuals and how those actions affect others.

As Demolombe and Lorini [15, 57] have shown, building a logical model of even part of a detailed trust model is a complex undertaking (they construct a modal logic model of one part of Falcone and Castelfranchi’s [25] concept of trust). Our focus is rather different. The aim of this paper is to begin to combine two lines of work — work on modelling trust, and work on reasoning using argumentation. While, as we discuss in the next section, there is a large amount of work on trust, we are most concerned with work on handling trust in social networks which deals with a simpler concept of trust than those we have discussed so far. Much of the work on trust in social networks, for example [37], has concentrated on identifying ways to take a social network in which people rate the trustworthiness of their acquaintances, and infer new ratings between pairs of individuals. Many models have been proposed, and their predictions compared with the ratings that people themselves generate. However, little work has been done on reasoning using these ratings.

Argumentation [73] on the other hand, is a field that is almost entirely concerned with reasoning and has a couple of features that we believe make it appropriate for dealing with trust. First, argumentation has been used to capture features of interaction between agents [67, 68], a situation in which trust needs to be considered, so it seems to be a natural step to think about how argumentation-based interactions might be extended with a model of trust. The first step in doing this is to develop a system of argumentation that can make use of trust information and that is what we do here. Second, argumentation provides a reason-based model of trust — it constructs arguments (reasons) for and against adopting beliefs and actions which explicitly record the agents that need to be trusted in the adoption. Thus our work contributes towards a model of trust that is reason-based as [11] argues trust should be. (The system we describe here is a system that constructs arguments that incorporate information about trust, not arguments about what

1 Bets can, of course, be expressed in terms of subjective probabilities [41].
information or agents to trust.) As we have suggested before [69], the fact that argumentation records the steps used in reaching conclusions makes it appropriate for reasoning where the provenance of information is important, as is widely acknowledged in handling trust [30,32].

In this paper, we develop a general system of argumentation that can represent trust information, and be used in combination with a trust network that captures relationships between agents. This system makes it possible for an agent to construct arguments where belief in the conclusions is related to the degree of trust in the agents who supplied the information that is used as premises in the arguments. We present several instantiations of the general system, and demonstrate that one of them gives intuitively appealing results on an illustrative example. Part of the novelty of the work is its fusion of trust and argumentation, a topic that has not received much attention (though we describe a couple of related papers in Section 9). Another part of the novelty comes from the fact that the model we develop is graphical. Though arguments, and the relationships between them, are often drawn as graphs, the underlying systems are not typically formulated graph-theoretically. Since one of our long-term aims is to use the graphical display of arguments as a way to help people make decisions in situations where trust — especially trust that stems from the provenance of information — is important, a graphical formulation seems natural. As part of this, we give a graphical formulation of the usual notions of acceptability [22], though with some slight variations from the notions [22].

The rest of the paper is structured as follows. Section 3 introduces models of trust and argumentation, and then Section 4 combines them. The combined model is an abstract model, and to make the discussion more concrete, Section 5 give some instantiations. Section 6 gives a large example, before Sections 7 and 8 consider the computational aspects of building the graphical models and propagating trust and belief values through them. Finally Section 9 reviews the contribution of the paper, particularly with respect to other work on trust and argumentation, and Section 10 concludes.

2 Related work

As we pointed out above, there has been much work in recent years that is concerned with trust, both within computer science and outside computer science. Much of the work on trust in computer science has concentrated on dealing with specific scenarios in which trust has to be established or handled in some fashion. Thus, for example, we see work on trust in peer-to-peer networks, including the EigenTrust algorithm [48] — a variant of PageRank [65] where downloads from a source play the same role as outgoing hyperlinks and which is effective in excluding peers who want to disrupt the network. [1] then builds on this, developing a mechanism that prevents peers manipulating their trust values to get preferential downloads. [96] is concerned with slightly different issues in mobile ad-hoc networks, looking to prevent nodes from getting others to transmit their messages while refusing to transmit the messages of others.

The internet, as the largest distributed system of all, is naturally a target of much of the research on trust. There have, for example, been studies on the development of trust in e-commerce through the use of reputation systems [78] and studies on how such systems perform [77, 87] and how such systems can be manipulated [54]. One interesting recent development is the idea of having individuals indemnify each other by placing some form of financial guarantee on transactions that others enter into [13, 14], thus providing a reputation mechanism that is strategy-proof. Another area of concern has to do with the reliability of sources of information on the web. [92], for example, investigates mechanisms to determine which sources to trust when faced with multiple conflicting sources, while [17] looks at the related question of how to resolve conflicting information, and [2] extends this idea to rate the individuals who provide information by looking at the history of the information they have provided. Issues related to trust in the social web have also attracted much attention [14, 62, 87, 91].

Trust is an especially important issue from the perspective of autonomous agents and multiagent systems. The premise behind the multiagent systems field is that of developing software agents that will work in the interests of their owners, carrying out their owners’ wishes while interacting with other entities. In such interactions, agents will have to reason about the amount that they should trust those other entities, whether they are trusting those entities to carry out some task, or whether they are trusting those entities to not misuse crucial information. As a result we find much work on trust in agent-based systems [81].

In the previous section, we mentioned the definition of trust suggested by Falcone and Castelfranchi [25], which relates trust explicitly to the goals of an agent, and considers trust to be concerned with whether another agent can and
will perform an action which will enable the first agent to achieve its goals. Much has been written on this model, and, from our perspective, \cite{11} and \cite{27} are the most important. \cite{11} argues that while trust has an element of “subjective certainty” to it, it is also reason-based, and this is a position that we completely agree with — as we will see below our handling of trust uses some numerical elements to capture aspects of that certainty or uncertainty while relying on argumentation to capture the reason.

\cite{27} discusses the issue of transitivity, which is often taken for granted in work on trust, but which always involves assumptions, assumptions which are analyzed in detail in \cite{27}.

Give the logical basis of the argumentation system that we develop, we should point out other work on logical approaches to trust. As we mentioned above, \cite{15, 57} have developed a logic that captures important elements of the definition of trust in \cite{23}. This model has also been extended \cite{9} to deal with situations in which the trusting agent has to rely on another agent for more than a single action. The focus on action, which is of course key in \cite{25}’s concept of trust, distinguishes these models from other logical accounts of trust, such as \cite{50} where the focus is on trust in information sources. This latter is much closer to our concern, though our notions of trust and the beliefs that are affected by them are much simpler than either those of \cite{15} and \cite{50}.

In work on trust in multiagent systems and the social web, it is common to assume that individuals maintain a trust network of their acquaintances, which includes ratings of how much those acquaintances are trusted, and how much those acquaintances trust their acquaintances, and so on. One natural question to ask in this context is what inference is reasonable in such networks, and the propagation of trust — both through the transitivity of trust relations \cite{50, 53, 79, 90} and more complex relationships like “co-citation” \cite{36} — have been studied. Our approach builds on this work by considering how agents can make use of information that comes from acquaintances for which a trust value is derived using this kind of computation. In this regard, the closest progenitor of the work in this paper is \cite{50}, which combines information from different individuals in a trust to reason about what one of these agents should do. (\cite{50} is also the paper from which we draw our ongoing example.)

\section{An abstract model of trust and argumentation}

In this section we present the formal models of trust and argumentation that we use in this paper.

\subsection{Some graph theory}

Since graphs of various kinds will crop up throughout this paper, we start with a little graph theory, based on the introductory chapters of \cite{16}. A graph is a pair \( G = (V, E) \) of sets. \( V \) is the set of vertices (or nodes). \( E \) is the set of edges (or arcs), and is a set of subsets of \( V \). Each element \( e \in E \) is the set of nodes joined by that edge. For much of this paper we will be concerned with graphs that include hyperedges, that is edges which join three or more vertices. Thus if \( V = \{v_1, v_2, v_3, v_4, v_5\} \), \( E \) might be \( \{\{v_1, v_2\}, \{v_3, v_4\}, \{v_1, v_3, v_5\}\} \). A graph that includes hyperedges is called a hypergraph. If \( V' \subseteq V \) and \( E' \subseteq E \) then \( G' = (V', E') \) is a subgraph of \( G \), which we write as \( G' \subseteq G \).

Many of the graphs we use will be directed graphs. A directed graph is a graph in which each edge has an initial vertex (or set of vertices) and a terminal vertex (or set of vertices). Thus for:

\[ E = \{\{v_1, v_2\}, \{v_3, v_4\}, \{v_1, v_3, v_5\}\}, \]

\( v_1 \) might be the initial vertex and \( v_2 \) the terminal vertex of the first edge, and \( v_1, v_3 \) might be the initial vertices of the third (hyper) edge and \( v_5 \) the terminal vertex. (The third edge might also have \( v_1 \) as the initial vertex and \( v_3, v_5 \) as the terminal vertices.)

A path in an undirected graph \( G \) is just a subgraph of \( G \) for which \( V = \{v_0, v_1, v_2, \ldots, v_k\} \) and \( E = \{\{v_0, v_1\}, \{v_1, v_2\}, \{v_{k-1}, v_k\}\} \). Where this is unambiguous, we will write the path as \( v_0, v_1, v_2, \ldots, v_k \). Clearly \( E \) has to be such that there is an edge joining every subsequent pair of vertices in \( V \). When the graph is directed, we distinguish between an undirected path, as defined above, and a directed path which obeys an additional constraint on path, that for every \( v_i \) and \( v_{i+1} \) in \( V \), \( v_i \) is the initial vertex of the edge that joins \( v_i \) and \( v_{i+1} \), and \( v_{i+1} \) is the terminal vertex. When we consider paths in hypergraphs, the constraint is that for every pair of vertices in the sequence, there is an edge in \( E \) that contains both

\footnote{The arguments that we consider here are admittedly not arguments for and against trusting another agent, but that aspect can be captured by the related system we describe in \cite{28}.}
vertices, and if that hypergraph is directed, the first edge in the pair must contain one of the initial vertices, and the second must contain one of the terminal vertices of the edge in question.

A connected graph is one for which there is a path between every pair of vertices. If the graph is directed, it is connected if there is an undirected path between every pair of vertices. A cycle is an (undirected) path \( v_0, \ldots, v_k, v_0 \). A graph that doesn’t contain any cycles is called a forest, and a forest that is connected is called a tree. The usual graph-theoretic terminology calls any vertex which is a member of only one edge a leaf. In the directed graphs we use, we will place an additional constraint on leaves. A leaf is a vertex that is part of only one edge, and which is the terminal vertex of that edge. In contrast, a root is a vertex that is part of only one edge, and which is the initial vertex of that edge.

### 3.2 Trust Networks

Given this graph theory, we can now describe the first important set of concepts that we will be dealing with. We are interested in a set of Agents \( \text{Ags} \) and the relationships between them, relationships that we assume are captured in a structure similar to a social network. In particular, we are interested in how the agents trust one another, and to do this we use a simple model of trust that is similar to that used in other work on social networks \([33, 50, 55]\). Following the usual presentation, for example \([36, 50, 53, 79, 90]\), we start with a trust relation:

\[
\tau \subseteq \text{Ags} \times \text{Ags}
\]

which we can think of as identifying which agents trust one another. If \( \tau(Ag_i, Ag_j) \) for \( Ag_i, Ag_j \in \text{Ags} \), then \( Ag_i \) trusts \( Ag_j \). Note that this is not a symmetric relation, so it is not necessarily the case that \( \tau(Ag_i, Ag_j) \Rightarrow \tau(Ag_j, Ag_i) \).

It is natural to represent this trust relation as a directed graph, and we have:

**Definition 1** A trust network for a set of agents \( \text{Ags} \) is a pair

\[
\mathcal{T} = \langle \text{Ags}, \{\tau\} \rangle
\]

where \( \{\tau\} \) is the set of pairwise trust relations over the agents in \( \text{Ags} \) so that if \( \tau(Ag_i, Ag_j) \) is in \( \{\tau\} \) then \( \{Ag_i, Ag_j\} \) is a directed arc in \( \mathcal{T} \).

In this graph, the set of agents is the set of vertices, and the trust relations define the arcs. We are typically interested in minimal trust networks, which we define to be trust networks that are connected. These thus capture the relationship between a set of agents all of whom, in one way or another, have something to say about trust in other members of the group. A directed path between agents in the trust network indicates that one agent directly or indirectly trusts others to which it is connected directly or indirectly. For example:

\[
\langle Ag_1, Ag_2, \ldots, Ag_n \rangle
\]

is a path from agent \( Ag_1 \) to \( Ag_n \), which requires that:

\[
\tau(Ag_1, Ag_2), \tau(Ag_2, Ag_3), \ldots, \tau(Ag_{n-1}, Ag_n)
\]

and looking at the path gives us a means to compute the trust that \( Ag_1 \) has in \( Ag_n \) given the trust that each agent along the path has in the next agent along the path. The usual assumption here is that we can place some measure of opinion on the trust that one agent has in another, so we have:

\[
tr : \text{Ags} \times \text{Ags} \mapsto O
\]

where \( O \) denotes the set of possible values of those opinions. A popular instantiation of opinion in the literature is the set of real numbers \( \mathbb{R} \):

\[
tr^\mathbb{R} : \text{Ags} \times \text{Ags} \mapsto \mathbb{R}
\]

where \( tr \) gives a suitable trust value. For example, we might take a value of 0 to indicate that there is no trust between the agents in question and a value of 1 to indicate the fullest possible degree of trust between the agents. Other instantiations of \( O \) might contain more structure, such as the tuple of numbers used in subjective logic \([45, 64, 90]\).
We assume that $tr$ and $\tau$ match, so that:

$$tr(A_g, A_g) \neq \perp \iff (Ag, A_g) \in \tau$$

$$tr(A_g, A_g) = \perp \iff (Ag, A_g) \notin \tau$$

where $\perp$ is the value from $\mathbb{R}$ that denotes no trust.

Now, this just deals with the direct trust relations encoded in $\tau$. It is usual in work on trust to consider performing inference about trust by assuming that trust relations are transitive. This is easily captured in the notion of a trust network:

**Definition 2** If $Ag_i$ is connected to $Ag_j$ by a set of directed paths $\{Ag_i, Ag_{i+1}, \ldots, Ag_j\}$ in the trust network $T$, then $Ag_i$ trusts $Ag_j$ according to $T$.

In other words, here we take trust to be a transitive notion. We note that while [27] argue that transitivity does not always hold — and we believe that it is absolutely correct that transitivity only holds under certain conditions — transitivity is an assumption that is commonly made in the literature. For example, the notion of trust embodied here is exactly the “indirect trust” or “derived trust” of [46], and the process of inferring this indirect trust is what [36] calls “direct propagation”. Our purpose here is to model those situations in which it is reasonable to consider trust to be transitive, while acknowledging that this is not always the case.

If we have a trust function $tr$, then we can compute:

$$tr(A_g, Ag_j) = tr(A_g, Ag_{i+1}, Ag_{i+2}, \ldots, Ag_{j-1}, Ag_j)$$

$$= tr(A_g, Ag_{i+1}) \otimes tr(A_{i+1}, Ag_{i+2}) \otimes \ldots \otimes tr(A_{j-1}, Ag_j)$$

for some function

$$\otimes : \mathcal{O} \times \mathcal{O} \mapsto \mathcal{O}$$

Here we follow [90] in using the symbol $\otimes$ to stand for this operation, while allowing it in practice to be one of a number of possible operations as we will discuss below. Sometimes it is the case that there are two or more paths through the trust network between $Ag_i$ and $Ag_j$ indicating that $Ag_i$ has several opinions about the trustworthiness of $Ag_j$. If we have two paths:

$$(Ag_i, Ag_{i+1}, \ldots, Ag_{j-1}, Ag_j)$$

$$(Ag_i, Ag_{i+1}, \ldots, Ag_{j-1}, Ag_j)$$

and

$$tr(A_g, Ag_j)' = tr(A_g, Ag_{i+1}) \otimes tr(A_{i+1}, Ag_{i+2}) \otimes \ldots \otimes tr(A_{j-1}, Ag_j)$$

$$tr(A_g, Ag_j)'' = tr(A_g, Ag_{i+1}) \otimes tr(A_{i+1}, Ag_{i+2}) \otimes \ldots \otimes tr(A_{j-1}, Ag_j)$$

then the overall degree of trust that $Ag_i$ has in $Ag_j$ is:

$$tr(A_g, Ag_j) = tr(A_g, Ag_j)' \oplus tr(A_g, Ag_j)''$$

again using the standard notation

$$\oplus : \mathcal{O} \times \mathcal{O} \mapsto \mathcal{O}$$

for a function that combines trust measures along two distinct paths between the same two vertices [90]. This approach can clearly be extended to deal with three or more paths. The literature contains several instantiations of $\otimes$ and $\oplus$. For example, [79] discusses using multiplication or minimum for $\otimes$ and using maximum for $\oplus$, while [50] uses a weighted average that in essence adopts multiplication for $\otimes$ and addition for $\oplus$. [46] and [90] use operators derived

---

3Indeed, in other work [70], we discuss explicitly modelling the ways in which trust is propagated, which makes it possible to perform metareasoning about which forms are applicable in different situations.

4[90] uses the unadorned symbol $\otimes$ and we add the subscript to distinguish it from the similar operation for belief values we introduce below.
from Dempster-Shafer theory, and as [8] discuss, we could instantiate operators like these in many other ways. We will discuss some of these options more below.

As an example of a trust graph, consider Figure 1(a) which shows the trust relationship between John, Mary, Alice, Jane and Dave. This is adapted from the example in [50] normalizing the values to lie between 0 and 1. The solid lines are direct trust relationships, the dotted lines are indirect links derived from the direct links. Thus John trusts Jane and Dave because he trusts Mary and Mary trusts Jane and Dave. However, John does not, even indirectly, trust Alice.

Since we will often be considering the viewpoint of a given agent, it is useful for us to define the concept of an agent-centric trust graph:

**Definition 3** An agent-centric trust graph $T$ is a trust network with a single root.

The agent-centric trust graph with $Ag$ at its root is said to be the network from $Ag$’s point of view or the “$Ag$-centric network”. Thus Figure 1(b) is the John-centric version of the graph in Figure 1(a) and Figure 1(c) is the Alice-centric version. We use this terminology because:

**Proposition 1** For an agent-centric trust network $T$ with $Ag$ at the root, $Ag$ trusts every agent in the network according to $T$.

**Proof:** An agent-centric trust network has just one root and so it is a tree. Every tree is connected, and in a directed tree such as $T$, there is a directed path from the single root to every node (if there was not, there would be more than one root). Thus a directed path connects $Ag$ to every other agent in $T$ and by Definition 2 $Ag$ trusts every agent according to $T$. $\square$

Thus an agent-centric trust network identifies the agents that are, direct and indirectly, trusted by the agent corresponding to the root, and every agent in the agent-centric trust network is trusted by the agent at the root. In addition, we can immediately see that:

**Proposition 2** For any trust network $T = \langle Ags, \{\tau\} \rangle$, there exists a distinct agent-centric trust network $T'$ for each node in $T$, and every $T'$ will be a sub-graph of $T$.

**Proof:** For the first part, we can construct a trust network $T' = \langle Ags', \{\tau'\} \rangle$ for every agent $Ag_i \in Ags$ by recursively identifying the agents $Ag_i$, it is linked to by directed arcs from $T$, and every agent that those $Ag_i$ are linked to, and so on. (If there are no outgoing arcs from an $Ag_i$, the graph will just include that one agent). Since these graphs all have
different root nodes, they are distinct. For the second part, it is clear that during this construction process, \( \text{Ags} \subseteq \text{Ags} \) and \( \{\tau\}' \subseteq \{\tau\} \), so \( T' \subseteq T \). □

**Proposition 3** Given a trust network \( T \) that contains \( \text{Ag}_i \) and \( \text{Ag}_j \) and an \( \text{Ag}_i \)-centric trust network \( T' \) such that \( T' \subseteq T \), \( \text{Ag}_i \) trusts \( \text{Ag}_j \) according to \( T' \) iff \( \text{Ag}_i \) trusts \( \text{Ag}_j \) according to \( T \).

**Proof:** From Proposition 3 \( T' \subseteq T \). As a result, every arc in \( T' \) is also an arc in \( T \), and so every path in \( T' \) is a path in \( T \). Thus \( \text{Ag}_i \) trusts \( \text{Ag}_j \) according to \( T' \) only if \( \text{Ag}_i \) trusts \( \text{Ag}_j \) according to \( T \). For the “if” part, again consider the recursive construction of \( T' \) sketched in the proof of Proposition 3. If there is a directed path from the root of \( T' \) to some agent in \( T' \), then that path will be in \( T' \) and so \( \text{Ag}_i \) trusts \( \text{Ag}_j \) according to \( T' \) for every \( \text{Ag}_j \) that it trusts according to \( T' \) only if \( \text{Ag}_i \) trusts \( \text{Ag}_j \) according to \( T' \).

Thus an \( \text{Ag}_i \)-centric trust network that is derived from a trust network \( T \) exactly identifies the agents that, according to \( T \), \( \text{Ag}_i \) trusts. As a result, when we consider what \( \text{Ag}_i \) reasons about, we lose nothing by ignoring the parts of \( T \) that aren’t in the \( \text{Ag}_i \)-centric network — they only contain agents that \( \text{Ag}_i \) can safely ignore because they aren’t trusted.

### 3.3 Argumentation

Now we turn our attention to the structure of the agents. An agent \( \text{Ag}_i \) has a knowledge base \( \Sigma = P \cup \Delta \). \( P \) is a set of premises, each of which is a logical statement in a language \( \mathcal{L} \). \( \Delta \) is a set of inference rules, which we individually denote by \( \delta \), each of which is of the form:

\[
\delta = \frac{\{p_1, \ldots, p_n\}}{c}
\]

where every \( p_i \) (denoted by \( p_i(\delta) \)) and \( c \) (denoted by \( c(\delta) \)) are members of \( \mathcal{L} \). In other words the inference rules link some set of premises to a conclusion. We will also write these rules as \( \langle \delta, c \rangle \). In this paper, since here we draw heavily on the work of [50], these rules are much like the normal default rules used in that work. For example, a domain specific inference rule on not watching a comedy film can be represented:

\[
\delta = \frac{\text{Comedy}(x)}{\neg \text{Watch}(x)}
\]

wedded to the use of these default-like rule. \( \Delta \) can be any set of inference rules, for example we might use the natural deduction style rules of [59].

We can represent inference using the rules in \( \delta \), whatever form they take, as another graph as follows:

**Definition 4** A rule network \( R \) is a directed hypergraph \( \langle V', E' \rangle \) where:

1. the set of vertices \( V' \) are elements of \( \mathcal{L} \);
2. the set of edges \( E' \) are inference rules \( \delta \);
3. the initial vertices of an edge \( e \in E' \) are the premises of the corresponding rule \( \delta \); and
4. the terminal node of that edge is the corresponding conclusion \( c \).

Thus a rule network simply connects premises and conclusions of rules. For a simple rule network, see Figure 2 (again this is taken from [50]).

Under certain circumstances, a rule network captures a proof made using the rules and premises in a particular \( \Sigma \):

**Definition 5** For a given knowledge base \( \Sigma = P \cup \Delta \), a rule network \( R = \langle V', E' \rangle \) is a proof network if and only if every premise of each \( \delta \in E' \) is either a member of \( P \) or the conclusion of some \( \delta' \in E' \).
Figure 2: A rule network. The rectangular nodes denote premises and the oval, which represent a hyperedge, denotes an inference rule.

To be a proof network, the rule network has to be constructed from the contents of some knowledge-base — a rule can’t be in the proof network unless its premises are either in the knowledge base or are derived by applying rules to premises that are in the knowledge base.

We say that a proof network is for a conclusion $c$ if $c$ is a leaf of the network. For example, if:

$$
\Sigma = \{ \text{IndieFilm}(hce), \text{DirectedBy}(hce, Almodovar) \} \\
\cup \{ \text{IndieFilm}(x) \land \text{DirectedBy}(x, Almodovar) \} \quad \frac{\text{Watch}(x)}{\text{Watch}(hce)}
$$

then Figure 2 is a proof network for Watch(hce).

Some proof networks correspond to arguments:

**Definition 6** An argument $A$ from a knowledge base $\Sigma = P \cup \Delta$ is a pair $\langle h, H \rangle$ where $H = \langle V', E' \rangle$ is a proof network for $h$, and $h$ is the only leaf of $H$.

$H$ is the support of the argument, and $h$ is the conclusion. $C(H)$ is the set of intermediate conclusions of $H$, the set of all the conclusions of the $\delta \in E'$ other than $h$. $P(H)$ is the set of pure premises of $H$, the premises of the $\delta \in E'$ that aren’t intermediate conclusions of $H$.

Thus for us an argument is a graphical representation of Prakken’s idea of an argument [72], where the argument contains a full trace of the steps that were used to reach the conclusion (rather than just containing the set of rules that were used as in [29]).

Given these definitions, we can easily show that:

**Proposition 4** If $\langle h, H \rangle$ is an argument, then $H$ is a connected graph.

*Proof:* The proof network of an argument is allowed only one leaf. Since it is a proof network, the leaf has to be a conclusion of a rule whose premises are either part of $\Sigma$, and thus roots of the graph, or the conclusions of other rules. Working backwards from the leaf/conclusion, it is clear that there can be no components of the graph that aren’t connected to the leaf, and so the graph must be connected. $\square$

**Proposition 5** If $\langle h, H \rangle$ is an argument, then there is only one hyperedge $\langle \delta, h \rangle$ in $H$.

*Proof:* This follows directly from the fact that $h$ is a leaf of $H$. If there were more than one rule in $H$ with $h$ as its conclusion, $h$ would not be a leaf, which contradicts Definition 6. $\square$

Thus an argument does not have any duplicated reasoning that supports its conclusion, and an argument is a proof network that is minimal in the sense that it sanctions no inferences other than its conclusion and intermediate steps that become premises of rules required to generate the conclusion.
Since our agents are often going to deal with information that is uncertain, we will assume that each agent $A_g$ has a function which assigns a degree of belief — which might be established as described in [58] — to elements of the logical language it uses for premises and rules:

$$\text{bel}_i : \mathcal{L} \rightarrow \mathcal{B}$$

where $\mathcal{B}$ denotes the set of possible values of the degree of beliefs. A common instantiation of $\mathcal{B}$ is the set of real numbers $\mathbb{R}$:

$$\text{bel}_i : \mathcal{L} \rightarrow \mathbb{R}.$$  

Furthermore, as with trust, we will often use a function that returns degrees of belief between 0 and 1. As we build arguments, we need to combine the degrees of belief assigned to premises and rules. For a rule:

$$\delta = \frac{\{p_1, \ldots, p_m\}}{c}$$

each agent $A_g$ associates a degree of belief $\text{bel}_i(\delta)$ with the rule along with a belief combination function:

$$\delta^i \otimes^c (\text{bel}_i(p_1), \ldots, \text{bel}_i(p_m), \text{bel}_i(\delta))$$

of the appropriate arity to combine the beliefs in all the premises with the belief in the rule itself to establish the appropriate belief in the conclusion. combination function $\delta^i \otimes$ can be tailored to reflect different belief combinations for different inference schemes (such as the modus ponens, modus tollens, and hypothetical syllogism [52]).

With a belief function $\text{bel}_i$ for each element of the grounds of an argument, we can assign belief to an argument as a whole:

$$\text{bel}_i([h, H]) = \text{combine}_i(\{\text{bel}_i(h_k | h_k \in H)\})$$

where $\text{combine}_i$ is a function to combine the belief on individual component of $H$ into the belief on the argument as a whole. Ignoring the internal structure of an argument, a simplified implementation of $\text{combine}_i$ is

$$\text{combine}_i(\{h_1, h_2, \ldots, h_n\}) = h_1 \otimes^\text{bel}_i h_2 \otimes^\text{bel}_i \ldots \otimes^\text{bel}_i h_n,$$

where:

$$\otimes^\text{bel}_i : \mathcal{B} \times \mathcal{B} \rightarrow \mathcal{B}$$

is a function for the pairwise combination of beliefs. Depending on the requirement of the application of our argumentation system, we can also implement the combination function by taking into account the inference structure of the arguments and applying the combination function $\delta^i \otimes$ of every inference rule used in the arguments.

A key notion in argumentation is that arguments defeat one another. That is, one argument casts doubt on another by, for example, contradicting the conclusion of the second argument. We distinguish a number of ways that a defeat may occur as follows:

**Definition 7** An argument $\langle h_1, H_1 \rangle$ defeats an argument $\langle h_2, H_2 \rangle$ if it rebuts, premise-undercuts, intermediate-undercuts, or inference-undercuts it, where:

- An argument $\langle h_1, H_1 \rangle$ rebuts another argument $\langle h_2, H_2 \rangle$ iff $h_1 \equiv \neg h_2$.
- An argument $\langle h_1, H_1 \rangle$ premise-undercuts another argument $\langle h_2, H_2 \rangle$ iff there is a premise $p \in P(H_2)$ such that $h_1 \equiv \neg p$.
- An argument $\langle h_1, H_1 \rangle$ intermediate-undercuts another argument $\langle h_2, H_2 \rangle$ iff there is an intermediate conclusion $c \in C(H_2)$ such that $c \neq h_2$ and $h_1 \equiv \neg c$.
- An argument $\langle h_1, H_1 \rangle$ inference-undercuts another argument $\langle h_2, H_2 \rangle$ iff there is an inference rule $\delta \in \Delta(H_2)$ such that $\delta = \frac{p_1, \ldots, p_n}{c}$ and $h_1 \equiv \neg (p_1 \land \ldots \land p_n \rightarrow c)$.

In any case in which $\langle h_1, H_1 \rangle$ defeats $\langle h_2, H_2 \rangle$, $\langle h_1, H_1 \rangle$ is said to be a defeater of $\langle h_2, H_2 \rangle$, and $\langle h_2, H_2 \rangle$ is said to be the defeatee. The relation defeat collects all pairs $\langle \langle h_1, H_1 \rangle, \langle h_2, H_2 \rangle \rangle$ such that $\langle h_1, H_1 \rangle$ defeats $\langle h_2, H_2 \rangle$. 

10
From Dung \cite{23}, we have the following component definitions, all of which hold for the system of argumentation we have described.

**Definition 8** An argumentation framework is a pair, \( \text{Args} = (A, R) \), where \( A \) is a set of arguments, and \( R \) is the binary relation over the arguments.

**Definition 9** Let \( (A, R) \) be an argumentation framework, and \( S \subseteq A \). An argument \( A \) is defended by \( S \) iff \( \forall B \in A \) if \( (B, A) \in R \) then \( \exists C \in S \) such that \( (C, B) \in R \).

**Definition 10** \( S \subseteq A \). \( FR(S) = \{ A \in A | A \text{ is defended by } S \text{ with respect to } R \} \).

Now, for a function \( F : D \rightarrow D \) where \( D \) is the domain and the range of the function, a fixed point of \( F \) is an \( x \in D \) such that \( x = F(x) \). When the \( D \) is associated with an ordering \( P \) — for example, \( P \) can be set inclusion over the power set \( D \) of arguments — \( x \) is a least fixpoint of \( F \) if \( x \) is a least element of \( D \) with respect to \( P \) and \( x \) is a fixed point.

**Definition 11** Let \( (A, R) \) be an argumentation framework. The set of acceptable arguments, denoted by \( \text{Acc}^F \), is the least fixpoint of the function \( FR \) with respect to set inclusion. The least fixpoint semantics can be viewed as a mathematical translation of the principle that an argument survives if it can defend itself and be defended by a set of arguments which can also survive all the attacks made upon its members. Other argumentation semantics from the literature, such as \[7, 12, 23\] can also be employed.

### 4 Arguments and Trust

So far we have described how trust is propagated between agents, and how each agent builds arguments. We now combine the two.

#### 4.1 Trust and belief

As we discussed above, we make use of a very simple model of belief, since for the purposes of this paper we only need to capture the fact that an agent’s knowledge can be uncertain. We capture this by allowing each agent \( Ag_i \) to have a measure of belief \( bel_i(\phi) \) for each \( \phi \) in its knowledge base \( \Sigma_i \). However, an agent \( Ag_i \) is not only interested in using information from its own knowledge base, but also information from other agents — for example let’s imagine that \( Ag_i \) is using \( \phi \) which \( Ag_j \) told \( Ag_i \) was true. \[50\] handles this situation by saying that if \( Ag_i \) trusts \( Ag_j \), and \( Ag_j \) says \( \phi \) is true, then \( Ag_i \) believes \( \phi \). We follow this principle but adapt it for our model where both trust and belief admit degrees.

To do this, we say that \( Ag_i \) needs to take into account its degree of trust in \( Ag_j \) in formulating its degree of belief in information it gets from \( Ag_j \). We assume that this is done by using the trust value that \( Ag_i \) can compute for \( Ag_j \) through the trust network that joins them, as outlined in Section 3.2. Having computed this value, which we will call \( tr(Ag_i, Ag_j) \), we further assume that \( Ag_i \) can convert this value into a degree of belief that it can use in argumentation, thus assuming a function \textit{trust-to-belief}:

\[
ttb : \mathbb{R} \mapsto \mathbb{R}
\]

that can take any trust value and map it to the correct degree of belief. Depending on the semantics of the degrees of trust and belief, this function may be the identity — for example if the trust values \( Ag_i \) has for every \( Ag_j \) is simply \( Ag_i \)’s subjective degree of belief that what \( Ag_j \) says is true, which is the notion of trust in \[28\, 63\] — though different semantics for trust and belief would require more complex transformations. \[95\] suggests translations between some pairs of belief measures which might be appropriate.) In any case, for \( Ag_i \), its belief in \( \phi \) may be computed:

\[
bel_i(\phi) = ttb(tr(Ag_i, Ag_j))
\]

In other words the belief that \( Ag_i \) has in \( \phi \) is a function of the trust that \( Ag_i \) has in \( Ag_j \). What we have so far assumes that \( Ag_j \) expresses the opinion that \( \phi \) is true. If \( Ag_j \) expresses some degree of belief in \( \phi \), then \( bel_j(\phi) \), \( Ag_i \) can compute its degree of belief in \( \phi \):

\[
bel_i(\phi) = ttb(tr(Ag_i, Ag_j)) \odot bel_j(\phi)
\]

In other words, the belief that \( Ag_i \) has in \( \phi \) is a combination of \( Ag_j \)’s belief in \( \phi \) and the trust that \( Ag_i \) has in \( Ag_j \).
4.2 Trust-extended argumentation

Since we believe that it is useful to keep track of where different pieces of information came from, for example in case the trust values in our trust network change, we find the concept of the trust-extended proof network to be useful. A trust-extended proof network marries trust networks and argument graphs. This concept is formally defined below, and an example is given in Figure 3.

Definition 12 A trust-extended proof network \( R_t \) is a pair \( \langle T, R \rangle \) of an agent-centric trust network \( T \) and a proof network \( R \) such that every rule \( \delta_i \) and every leaf \( p_j \) of \( R \) are connected to a node \( Ag_k \) in \( T \) by an arc \( \langle Ag_k, \delta_i \rangle \) or \( \langle Ag_k, p_j \rangle \) that denote, respectively, that \( \delta_i \in \Delta_k \) or \( p_j \in \Sigma_k \) respectively.

An example of a trust-extended proof network is given in Figure 3. This shows how the acquaintances of \( Ag_1 \) each provide the elements of an argument. \( Ag_2 \) and \( Ag_5 \) each provide a premise of the rule \( \delta_4 \), while \( Ag_7 \) provides the rule itself, and together these support the conclusion \( \xi \). The idea behind the trust-extended proof network is that it relates the premises of an argument to their sources. Such a network therefore captures the reasoning of the agent at the root of the trust network, including which pieces of information it has used from which agents it trusts. It is simple to show that:

Proposition 6 A trust-extended proof network has one root and possibly many leaves, but only one leaf that is a conclusion of the proof network.

Proof: A trust-extended proof network is proof-network where every root in the proof network is linked to a node in an agent-centric trust network. Since there is, by definition, only one root in the trust network, there is only one root in the trust-extended proof network. Equally, although there may be many leaves of the trust network that are not linked to elements of the proof network, the proof network by definition only has one conclusion, and this will therefore be the only conclusion of the trust-extended network. ☐

Thus a trust-extended proof network relates a single agent to a single conclusion, and we can easily extend the notion of an argument — which as we recall from Definition 6 is a pair consisting of a proof network and its conclusion — with the trust information of a trust-extended proof network:

Definition 13 A trust-extended argument \( A' \) from the union of a set of knowledge bases \( \{ \Sigma_1, \ldots, \Sigma_n \} \) belonging to a set of agents \( Ags = \{ Ag_1, \ldots, Ag_n \} \), all of which are in \( T \), is a pair \( \langle h, R' \rangle \) where \( R' \) is a trust-extended proof network for \( h \), and \( h \) is the only leaf of \( R' \).
In the same way that an argument is relative to a knowledge base, so a trust-extended argument is relative to a set of agents, and, in particular, to the set of knowledge bases of those agents. Furthermore, the conclusions of a trust-extended argument are relative to a specific agent. Given a trust-extended argument, the only agent that is sanctioned to infer the conclusion of the argument is the one at the root of the trust graph. Thus, like the graph, the conclusions are agent-centric. Figure 4 presents two trust extended arguments. The leftmost, in Figure 4a, contains the same information as a previous example, but now — in keeping with the spirit of trust-extended arguments — identifies the origin of the information. In this case it is John whose reasoning is being captured, and this agent has access to the information:

\[ \Sigma_{\text{john}} = \{ \text{IndieFilm}(hce), \text{SpanishFilm}(hce), \text{DirectedBy}(hce, \text{Almodovar}) \} \]

while John’s acquaintance Dave says that:

\[ \delta_{\text{dave}} = \frac{\text{IndieFilm}(x) \land \text{DirectedBy}(x, \text{Almodovar})}{\text{Watch}(x)} \]

allowing John to construct an argument for \( \text{Watch}(hce) \).

The second argument, in Figure 4b, makes use of information from another acquaintance, Jane, who holds that:

\[ \delta_{\text{jane}} = \frac{\text{IndieFilm}(x) \land \text{SpanishFilm}(x)}{\neg \text{Watch}(x)} \]

from which John can construct an argument for \( \neg \text{Watch}(hce) \).

The last element of our model is the trust-extended argument graph, in which we show the relations between a set of trust-extended arguments. Informally, this is a set of trust-extended arguments with the defeat relationships between the arguments denoted by labelled edges. There are four kinds of defeat edge, one for each of the kinds of link identified in Definition 7.

**Definition 14** A trust-extended argument graph \( AG' \) from the union of a set of knowledge bases \( \{ \Sigma_1, \ldots, \Sigma_n \} \) belonging to a set of agents \( \text{Ags} = \{ A_{i_1}, \ldots A_{i_n} \} \), all of which are in \( T \) is the union of a set of trust-extended arguments \( A'_1 \) with an additional set \( D \) of edges which we call defeat edges.

There are four kinds of defeat edge: a rebut edge, a premise-undercut edge, an intermediate-undercut edge and an inference-undercut edge. Each defeat edge \( d \in D \) links two arguments \( A'_1 = (h_i, R'_j) \) and \( A'_j = (h_j, R'_i) \).

- \( A'_i \) and \( A'_j \) will be joined by a rebut edge iff \( h_i \equiv \neg h_j \).
- \( A'_i \) and \( A'_j \) will be joined by a premise-undercut edge iff there is some rule \( R'_j \) with a premise \( p \) and \( h_i \equiv \neg p \).
- \( A'_i \) and \( A'_j \) will be joined by an intermediate-undercut edge iff there is some rule \( R'_j \) with conclusion \( c \) and \( h_i \equiv \neg c \).
- \( A'_i \) and \( A'_j \) will be joined by an inference-undercut edge iff there is some rule \( R'_j \) with premises \( p_1, \ldots, p_n \) and conclusion \( c \), and \( h_i \equiv \neg (p_1 \land \ldots \land p_n \rightarrow c) \).

In any case in which \( A'_1 \) defeats \( A'_2 \), \( A'_1 \) is said to be a defeater of \( A'_2 \), and \( A'_2 \) is said to be the defeatee.

This definition clearly mirrors Definition 7 and with good reason. A trust-extended argument graph is a graphical counterpart of the argumentation framework of Definition 8 and we can identify sets of acceptable arguments from the graph just as we can from an argumentation framework.

Given the equivalence of the graphical framework we introduce here and previous non-graphical frameworks, a reasonable question is why we bother with the graphical framework. There are two answers. One is that the graphical approach links neatly with trust networks, and gives us a uniform representation to use as the basis for computations involving argumentation and trust — we describe how this may be done in Sections 7 and 8. The second answer is that our long-term goal is to use the framework we describe as the basis for a decision support tool that helps human users to understand the impact of trust on their decisions. Previous work on graphical representation of arguments, for example [10, 49, 66, 78, 80, 88, 89], suggests that a graphical representation will be a good way to do this, and here we have a graphical representation that can exactly mirror the computation of acceptability of arguments.
\[ \delta_{\text{dave}} = \text{IndieFilm}(\text{dave}) \land \text{DirectedBy}(\text{dave}, \text{Almodovar}) \land \text{Watch}(\text{dave}) \]

\[ \delta_{\text{jane}} = \text{IndieFilm}(\text{jane}) \land \text{Watch}(\text{jane}) \]

Figure 4: Two trust-extended arguments that John can construct
An example of a trust-extended argument graph is shown in Figure 5. This particular graph, which corresponds to the example we have discussed throughout this paper (and which we will continue to work with) shows the two arguments that John can develop (previously seen as separate trust-extended arguments in Figure 4) as one trust-extended argument graph. As before, the section in the middle of the graph is a trust network which shows the relationship between John, Mary, Dave and Jane. The two sections on either side of the trust network show arguments that John can develop. Each of these uses information from John’s knowledge (the rectangular nodes linked to the john node in the trust graph) and an inference rule from one of John’s acquaintances (the oval nodes linked to the dave and jane nodes in the trust graph). The conclusions of these two arguments (the rectangular nodes at the bottom of the graph) are joined by two rebut edges since the two arguments rebut each other, disagreeing with each other’s conclusions.

5 Instantiations of the abstract model

So far, our discussion of the trust models, and the way that the trust information is used in argumentation, has been rather abstract. In this section, we make the discussion more concrete by describing some instantiations of the model. In particular, in this section we discuss operations that might be used for $\oplus^{tr}$ and $\otimes^{tr}$ in combining trust values.

5.1 Possibility theory

As suggested by [79], possibility theory [18, 19, 20] is a natural way to interpret trust values. In possibility theory, itself derived from the notion of fuzzy sets [93, 94], numbers are attached to propositions to quantify the degree of uncertainty in the propositions, but the values are not probabilities. Instead, they are taken to represent the degree of possibility that the proposition is true. The possibility of a proposition $p$ is written as $\Pi(p)$.

Informally, the degree of possibility can be taken to suggest how surprised one would be to discover that the proposition is true. A proposition with a degree of possibility of 1 is a proposition that is thought to be completely possible, so one that would occasion no surprise were it found to be the case. A proposition with a lower degree of possibility would create some surprise were it found to be true.

There are several families of functions that may be used to combine possibility values, but the most commonly used are the following. The possibility of a conjunction $p \land q$ is taken to be greater than or equal to the minimum of the possibility of $p$ and the possibility of $q$:

$$\Pi(p \land q) \geq \min(\Pi(p), \Pi(q))$$

and so if we capture trust using the lower bound of the possibility values, then $\otimes^{tr}$ is $\min$. Similarly the possibility of a disjunction $p \lor q$ is typically taken to be the maximum of the possibilities of $p$ and $q$:

$$\Pi(p \lor q) = \max(\Pi(p), \Pi(q))$$

and so using possibility theory to model trust it is natural to take $\oplus^{tr}$ to be $\max$.

5.2 Subjective logic

The Dempster-Shafer theory [82, 83] is a generalization of probability theory (and, indeed, possibility theory [21]) in which probabilities can be assigned not just to individual propositions, but also to sets of propositions. This, it is argued, allows for the expression of a degree of ambiguity — if we have evidence that suggests either $p$ or $q$ is the case we can assign probability to the set $\{p, q\}$ and wait to see if additional evidence allows us to refine this opinion. A typical way to make use of this ability is to assign probability to some proposition $p$, its negation $\neg p$ and the set $\{p, \neg p\}$, with the latter representing the belief that one cannot, for lack of evidence, assign to either $p$ or its negation. This can be interpreted as the degree to which one cannot tell whether $p$ is true or not.

Subjective logic [42, 43, 64] uses Dempster-Shafer theory in exactly this way, modelling the trust in every proposition as a triple, and this is the approach taken by [90]. In this latter work, the trust links are labeled by a belief measurement:

$$tr(A_g, A_j) = M_{i,j}$$
Figure 5: A trust-extended argumentation graph capturing John’s view of the film example
where \( M_{i,j} = (b, d, u) \) and \( b \), denoted by \( b(m_{i,j}) \), is the probability that what \( Ag_j \) tells \( Ag_i \) will be true is actually true; \( d \), denoted by \( d(m_{i,j}) \), is the probability that what \( Ag_j \) tells \( Ag_i \) is true is actually false; and \( u \), denoted by \( u(m_{i,j}) \), is the probability of uncertain outcomes (that \( Ag_j \), for example, will not know if what it is told is true or not). Subjective logic, with its triple \( M_{i,j} \) representing trust, is thus an example of a trust measure where opinions \( O \) are more complex than just a single number.

Wang and Singh [90] describe how these values can be linked to an agent’s experience so that the values \( b(m_{i,j}) \), \( d(m_{i,j}) \) and \( u(m_{i,j}) \) are determined by how often what agent \( j \) tells agent \( i \) turns out to be the case. In particular Wang and Singh consider that agents keep track of the number of positive outcomes (where something that they are told is true turns out to be false, or vice versa) for each agent they interact with. Then:

**Definition 15** The evidence space is defined as

\[
E = \{ (r, s) \mid r > 0, s > 0 \}
\]

where \( r \) is the number of positive outcomes and \( s \) is the number of the negative outcomes, and the belief space is defined as:

\[
B = \{ (b, d, u) \mid b > 0, d > 0, u > 0, b + d + u = 1 \}
\]

This just says that for each agent that \( Ag_i \) interacts with, there are positive and negative instances of interaction, and, separately, there is the triple measure \( (b, d, u) \) that allows the outcome of future interactions to be estimated. Clearly we want to derive from the former, and we do this as follows.

**Definition 16** Let \( Z = (B, D, U) \) be a transformation from \( E \) to \( B \) such that \( Z(r, s) = (B(r, s), D(r, s), U(r, s)) \) where:

\[
\begin{align*}
B(r, s) &= c(r, s) \frac{r + 1}{r + s + 2} \\
D(r, s) &= c(r, s) \frac{s + 1}{r + s + 2} \\
U(r, s) &= 1 - c(r, s)
\end{align*}
\]

and where \( c(r, s) \) is the certainty level computed by a statistical model of certainty given the evidence of the trust links.

Thus the belief space is computed rather straightforwardly from the history of interactions provided that we have a model of certainty. One model of certainty is to let

\[
c(r, s) = \frac{1}{2} \int_0^1 |f_{r,s}(x) - 1| dx
\]

where

\[
f_{r,s}(x) = \frac{x^r(1-x)^s}{\int_0^1 x^r(1-x)^s dx}
\]

\( f_{r,s}(x) \) is the posterior (cumulative) probability of the positive outcomes (of a binary event with \( x \) as its prior) after observing \( r \) positive outcomes and \( s \) negative outcomes. \( c(r, s) \) is then defined as mean absolute deviation.

Given this model, the operators \( \otimes^{tr} \) and \( \oplus^{tr} \) can be defined as the following.

**Definition 17** Suppose \( M_1 = (b_1, d_1, u_1) \) and \( M_2 = (b_2, d_2, u_2) \), then \( M = M_1 \otimes M_2 = (b, d, u) \) where

\[
\begin{align*}
b &= b_1 b_2 \\
d &= b_1 d_2 \\
u &= 1 - b_1 b_2 - b_1 d_2
\end{align*}
\]

\(^5\)Conceptually, at least, this uncertainty can later be resolved, and the probability or true or false increased accordingly.
**Definition 18** Suppose \( M_1 = (b_1, d_1, u_1) \) and \( M_2 = (b_2, d_2, u_2) \), and let \((r_1, s_1)\) and \((r_2, s_2)\) be the elements in the event space of \( M_1 \) and \( M_2 \) respectively, then \( M = M_1 \oplus M_2 = (b, d, u) \) where

\[
\begin{align*}
    b &= B(r_1 + r_2, s_1 + s_2) \\
    d &= D(r_1 + r_2, s_1 + s_2) \\
    u &= U(r_1 + r_2, s_1 + s_2)
\end{align*}
\]

5.3 TidalTrust

The two models we have discussed so far are taken from the literature of reasoning under uncertainty. In other words, using these models to represent and compute trust values is assuming, as suggested by [28][50], that trust in an individual is the subjective belief that the individual tells the truth. An alternative view of trust is discussed in [50], where the authors’ aim is to construct a model that agrees with the way that people compute with trust values. (The values themselves also differ — while possibility theory and subjective logic deal with values in the range \([0, 1]\), [50] uses values between 0 and 10.) The result is the TidalTrust model, which describes how to propagate values through the kind of trust network we described earlier.

In particular, in the context of a extended trust network \( \mathcal{T} = \langle Ag_s, \{\tau\} \rangle \), we can use the TidalTrust model to propagate trust along paths through the network by recursively computing:

\[
tr(Ag_i, Ag_j) = \sum_{(Ag_i, Ag_k) \in \mathcal{T} \text{ and } tr(Ag_i, Ag_k) \geq \theta tr(Ag_i, Ag_j)}^{tr'(Ag_i, Ag_k)} tr(Ag_k, Ag_j)
\]

where \( \theta \) is a parameter to ensure the model only considers trust values above a certain threshold thus ignoring small trust values, \( \oplus' \) is implemented as the arithmetic addition “+” and \( \odot' \) is implemented as the arithmetic multiplication “×”.

The above formula carries out a breath-first search through the trust network. To evaluate an agent \( Ag_i \)'s trust on another agent \( Ag_j \), the algorithm has each neighbor agent \( Ag_k \) of \( Ag_i \) obtain their evaluations of the trust to the agent \( Ag_j \) (in a same way), and then \( Ag_i \) computes the weighted average of its neighbors’ trust values on \( Ag_j \) with the weights being set to its trust values on each neighbor \( Ag_k \).

6 Examples

In this section we show how the formal system we have introduced can capture two versions of the the example from [50].

6.1 The original example

This example, from which we have been drawing throughout the paper, considers reasoning in the FilmTrust [26][34] database. This is an online database where people offer ratings for films they have seen and search for recommendations for films to watch. FilmTrust members also rate each other, indicating who they trust to give good advice on films. Although we have seen various pieces of the example already, we will start with a recap.

In the example, we are concerned with a certain agent John who is invited to watch a film by one of his friends. John is part of the trust network from Figure 1(a), and furthermore has the following information about the film in question:

\{IndieFilm(hce), SpanishFilm(hce), DirectedBy(hce, almodovar)\}

In the original example, John also has the rule:

\[
\delta_{john} = \frac{Comedy(x)}{-Watch(x)}
\]

6“Almodovar” here is Pedro Almodovar, and “hce” is an abbreviation for his 2002 film Hable con ella (Talk to her). It is, of course, arguable whether “hce” is an independent film, but since the original example considered it to be one, so will we.
Figure 6: A trust-extended argumentation graph capturing John’s view of the original film example, labelled with trust values
which indicates that he doesn’t like to watch comedies.

Since this information doesn’t help him to decide whether to watch the film, John asks people in his social network (for all of whom he can compute a trust rating) for their opinions and learns:

\[
\delta_{\text{jane}} = \frac{\text{IndieFilm}(x) \land \text{SpanishFilm}(x)}{\neg \text{Watch}(x)}
\]

\[
\delta_{\text{dave}} = \frac{\text{IndieFilm}(x) \land \text{DirectedBy}(x, \text{almodovar})}{\text{Watch}(x)}
\]

From this information, John can, as we have already discussed, construct two simple arguments which are depicted in Figure 6, and which we can translate as:

I should watch hec because it is an independent film and it is directed by Almodovar, and Dave says that any independent film directed by Almodovar is watchable.

and

I shouldn’t watch hec because it is a Spanish independent film, and Jane says that Spanish independent films are unwatchable.

These arguments rebut each other and John can use information about his friends in order to further refine his view of the arguments. However, to do that he has to choose a specific instantiation of the trust values. Let us assume he models trust using possibility theory. Then, as discussed in the previous section, \(\otimes\) is min. Considering the trust graph in Figure 1(a) it is easy for John to compute that:

\[
\text{tr}(\text{john}, \text{dave}) = 0.8
\]

\[
\text{tr}(\text{john}, \text{jane}) = 0.7
\]

and these are the values we see in Figure 6.

Now, these values need to be turned into belief values. Since the trust values are expressed using possibility theory it is natural to handle belief in the same way — this means that we handle the notion of strength of arguments in a way that is close to that adopted by [4] though the authors of that paper do not explicitly mention the use of possibility theory. Using the same values for trust and belief means that we interpret the degree of trust that one agent, \(A_g_i\), has in another, \(A_{g_j}\), to be the degree of belief that \(A_g_i\) has that what \(A_{g_j}\) says is true (exactly as in [28, 63]).

In other words the degree of belief that John has in a statement from Dave is exactly John’s degree of trust in Dave. As a result:

\[
\text{bel}_{\text{john}}(\delta_{\text{dave}}) = 0.8
\]

\[
\text{bel}_{\text{john}}(\delta_{\text{jane}}) = 0.7
\]

In order to reach conclusions about watching hec, these rules need to be combined with John’s initial knowledge, all of which he believes to be completely possible so that:

\[
\text{bel}_{\text{john}}(\text{IndieFilm}(\text{hec})) = 1
\]

\[
\text{bel}_{\text{john}}(\text{SpanishFilm}(\text{hec})) = 1
\]

\[
\text{bel}_{\text{john}}(\text{DirectedBy}(\text{hec}, \text{almodovar})) = 1
\]

Since these are possibility values, we use minimum for \(\otimes\), and so:

\[
\text{bel}_{\text{john}}(\text{Watch}(\text{hec})) = 0.8
\]

\[
\text{bel}_{\text{john}}(\neg \text{Watch}(\text{hec})) = 0.7
\]

We have not discussed how to use these belief values in conjunction with argumentation (we will come back to this topic below), but since as we mentioned, we are handling belief rather like Amgoud and Cayrol [4], we can take their approach to resolving defeat using belief. In other words, one argument defeats another if it defeats it in the sense of Definition 14 and its conclusion has a higher degree of belief. In this case, John will decide to watch hec.

While this is a simple example, it shows that our approach handles the combination of trust and argumentation in an intuitively appealing way, as well as agreeing with the analysis in [50].
6.2 The modified example

The previous example was included to show how our system can capture the reasoning from [50]. However, argumentation is typically less interested in the kind of symmetrical “rebut” conflict between the two arguments seen in that example than it is in the asymmetrical “undercut” conflict. The modified example will illustrate undercutting. We consider that John has a slightly different set of information:

\{SpanishFilm(hce), DirectedBy(hce, almodovar)\}

which does not include the information about hce being an indie film (as one reviewer pointed out, hce is not really an indie film). Rather he has the rule:

\[ \delta_{john} = \frac{DirectedBy(x, almodovar)}{¬IndieFilm(x)} \]

indicating that he believes that no Almodovar film is an indie film. We will further assume that, this time around, when John polls his social network about watching the film, he gets no reply from Dave, but does hear from Mary (who thinks hce is an independent film), and from Jane who replies as in the previous example. Thus John has the information that:

\[ \delta_{jane} = \frac{IndieFilm(x) \land SpanishFilm(x)}{¬Watch(x)} \]

\[ p_{mary} = \frac{IndieFilm(hce)}{\text{ }} \]

This time John has the arguments of Figure 7. Handling trust values as before — again using the approach from [4] — John will find that his argument against hce being an independent film defeats the argument for not watching hce. (Of course this leaves John with no information about whether to watch the film, he just knows that he doesn’t have an acceptable argument to not watch it).

7 Constructing trust-extended argumentation graphs

So far in this paper we have introduced a graphical representation of a variant of Dung’s argumentation framework which also captures information about which agents hold which pieces of information that are used to construct an argument, and what the trust relationship between the agents is. In this section we describe how to construct these trust-extended argumentation graphs. Here we separate the construction procedure (Algorithm 2) from the trust and belief propagation for clarity. However, in practice, we might wish to combine the two procedures together to avoid constructing unnecessary arguments.

From the algorithmic point of view, instead of constructing each proof network and the corresponding argument explicitly, we mark each proof network by its unique ID, and each arguments by its unique ID. The graph construction is based on the construction of proof networks in Algorithm 1 (below).

Proposition 7 Given a query q, Algorithm 1 creates all possible proofs of q. Each proof of q is marked with a unique ID, id \( \in \text{ID}(G, node(q)) \), and each proof with id is associated with a sub-proof network SubIDs(G, id).

Proof: Algorithm 1 goes through each inference rule \( \delta \in \Delta \) whose conclusion can be unified with q, and tries to construct a proof network for each premise of \( \delta \). If all premises have proof networks, the algorithm then connects node(q) to node(\( \delta \)), and node(\( \delta \)) to the nodes corresponding to the premises of \( \delta \). At the same time, the algorithm records each combination of the proofs for \( \delta \)'s premises as a proof network for q and marks it down with a unique ID. This unique ID is recorded into the set of proofs for q: ID(G, node(q)) □

In Algorithm 1, the ID set for a node node(q), denoted by ID(G, node(q)), records a set of arguments for q. Given an ID id \( \in \text{ID}(G, node(q)) \), the SubIDs(G, id) = \( \langle id_1, id_2, ..., id_k \rangle \) tracks the proof sub-networks identified by \( \langle id_1, id_2, ..., id_k \rangle \). With this ID construction, we can then recover each proof network of Definition 12 by traversing...
Figure 7: A trust-extended argumentation graph capturing John’s view of the modified film example, labelled with trust values
Proposition 8  Given a query $q$, Algorithm 2 creates an argumentation graph for $q$.

- For each node in the argument graph, if there are proof networks for it, all the proof networks are included in the argument graph; all the proof networks included are valid.
- For each node in the argument graph, if there are defeaters, the defeaters are included in the argument graph.
- If there is a defeat between two arguments, it is marked in $ID_{defeater, defeatee}$.

Proof: Algorithm 2 calls Algorithm 1 which constructs all the valid proof networks for each node. It also goes through all the nodes, and tries to construct proof networks and defeaters recursively with Algorithm 1 for all these nodes. □

In a similar manner to argument construction, Algorithm 2 records the IDs of arguments where one argument defeats another argument associated with two nodes $nd_1$ and $nd_2$ (which are either premise nodes, conclusion nodes or inference nodes) in the argument graph using $ID_{defeat}(nd_1, nd_2)$. The set of argument ID pairs which are recorded in $ID_{defeat}$ (for all nodes recorded) recovers the defeat relation in Definition 8. We can then use the defeat relation recorded in $ID_{defeat}$ to compute various argumentation semantics, such as the one in Definition 11. Following a similar approach to [3] the same defeat relation recorded in $ID_{defeat}$ can be refined with the trust and belief values, and in turn the argumentation semantics can be refined.

As Algorithm 1 and Algorithm 2 work in a dynamic programming manner, the complexity of these procedures is determined by the size of the syntactic structure of the knowledge base and the inference rule base, and the number of all possible arguments in the output.

Proposition 9  Assume that that rule base $\Delta$ only contains first order schemes (i.e. there are domain variables in the rules but no predicate variables). Let $P_\Delta = \cup_{\delta \in \Delta} \text{Premises}(\delta)$, $C_\Delta = \cup_{\delta \in \Delta} \{c(\delta)\}$ and the $K$ be the maximum arity of predicates and $M$ be the maximum number of values can be assigned to the variables of predicates. The number of reasoning links in the argumentation graph for a query $q$ created by Algorithm 2 is bounded by $O(K^2 \times M^2 \times (1 + |P_\Delta|) \times (|\Sigma| + C_\Delta))$. The number of defeat links is bounded by $O(K^2 \times M^2 \times (1 + |P_\Delta| + |\Sigma| + C_\Delta)) \times (|\Sigma| + C_\Delta))$.

Proof: The number of all possible conclusions and intermediate conclusions is $(|\Sigma| + |C_\Delta|) \times K \times M$, and the number of all possible premises is $(1 + |P_\Delta|) \times K \times M$. All possible reasoning links are between the set of all possible (intermediate) conclusions and the set of all possible premises, therefore the number of reasoning links is bounded by $O(K^2 \times M^2 \times (1 + |P_\Delta|) \times (|\Sigma| + C_\Delta))$. On the other hand, all possible defeat links are between the set of all possible conclusions and the union of all the possible premises and all the possible (intermediate) conclusions, therefore the number of defeat links is bounded by $O(K^2 \times M^2 \times (1 + |P_\Delta| + |\Sigma| + C_\Delta)) \times (|\Sigma| + C_\Delta))$. □

Note that, although we have an argumentation graph with size bounded polynomially in the number of premises the and number of inference rules, the number of arguments can still be exponential in the number of premises the and number of inference rules. We deliberately record this exponential number of arguments and their interactions in the form of IDs in Algorithm 1 and Algorithm 2. Part of our future research focuses on how to utilize this compact graphical structure for nearly tractable algorithms for trust and belief propagation with trust extended argumentation. In this work, we only consider first order reasoning rules with no function symbols. For second order rules, such as the natural deduction rules of [59] with predicate variables over the logical connectives, we will need an additional reasoning mechanism to efficiently instantiate the deduction rules to connect the unified formulae. This extension is also part of our planned future research.

A trust extended argumentation graph that can be created by Algorithm 2 from our running example is given in Figure 8. With IDs expanded into the corresponding arguments and centered on John’s view, we get the same trust extended argumentation graph as in Figure 5.
Algorithm 1: \textit{constructProofNet}(G, q, T, \Sigma, \Delta): Construct a proof network for \( q \)

\textbf{Input}: \( G \): a shared graph structure; \( q \): query; \( T \): trust network; \( \Sigma \): knowledge base; \( \Delta \): rule base

\begin{enumerate}
\item \textbf{if} node\((q)\) \textbf{in} \( G \) then
  \begin{enumerate}
  \item return \textit{node}(q);
  \end{enumerate}
\item \textbf{end}
\item \textbf{if} \( q \in \Sigma \) then
  \begin{enumerate}
  \item Create a new node \textit{node}(q) in \( G \);
  \item \textbf{for} each agent \( Ag_i \) such that \( q \) is in \( Ag_i \)'s knowledge base \( \Sigma_i \) do
    \begin{enumerate}
    \item Add a holding link \langle \textit{node}^T(Ag_i), \textit{node}(q) \rangle \text{ to } G; 
    \end{enumerate}
  \item \textbf{end}
  \item Allocate a new \textit{id};
  \item \( ID(G, \textit{node}(q)) \leftarrow \{id\} \);
  \item return \textit{node}(q);
  \end{enumerate}
\item \textbf{end}
\item \textbf{if} \texttextit{No inference rule } \( \delta \in \Delta \) with conclusion \textit{c(\delta)} \text{ unifiable with } \( q \) \text{ then}
  \begin{enumerate}
  \item return \( \emptyset \);
  \end{enumerate}
\item \textbf{end}
\item Create a new node \textit{node}(q) into \( G \);
\item \( ID(G, \textit{node}(q)) \leftarrow \emptyset \);
\item \textbf{for} each inference rule \( \delta \in \Delta \) with conclusion \( \textit{c(\delta)} \) \text{ unifiable with } \( q \) do
  \begin{enumerate}
  \item Compute a unifier \( \theta \leftarrow \text{unify}(q, \textit{c(\delta)}) \);
  \item \textbf{for} each rule premise \( p_i(\delta[\sigma]) \) \textbf{ do}
    \begin{enumerate}
    \item \textit{result}[p_i] \leftarrow \textit{constructProofNet}(G, p_i, T, \Sigma, \Delta);
    \end{enumerate}
  \item \textbf{end}
  \item \textbf{if} all \textit{result}[p_i]s are not empty \textbf{ then}
    \begin{enumerate}
    \item Refine the unifier \( \theta \) with the one produced for proof net \textit{result}[p_i]s;
    \item Create a new node \textit{node}(\( \delta[\theta] \)) in \( G \) if \textit{node}(\( \delta[\theta] \)) \notin \( G \);
    \item Add \langle \textit{node}(\( \delta[\theta] \)), \textit{node}(q) \rangle \text{ to } G;
    \item Add \langle \textit{p}_i, \textit{r}[\theta] \rangle \text{ to the } G \text{ for each } \textit{p}_i;
    \item \textbf{for} each agent \( Ag_i \) such that \( \delta \) is in \( Ag_i \)'s rule base do
      \begin{enumerate}
      \item Add a holding link \langle \textit{node}^T(Ag_i), \textit{delta}[\theta] \rangle \text{ to } G;
      \end{enumerate}
    \item \textbf{end}
    \end{enumerate}
  \item \textbf{end}
  \item /* Every combination of the proofs for premises is a new unique proof for \( q \) */
  \item \textbf{for} each combination \( (id_1, id_2, ..., id_k) \in \Pi, \text{ID}(G, \textit{node}(p_i)) \) do
    \begin{enumerate}
    \item Allocate a new \textit{id};
    \item \( ID(G, \textit{node}(q)) \leftarrow ID(G, \textit{node}(q)) \cup \{id\} \);
    \item SubIDs(G, id) \leftarrow \langle id_1, id_2, ..., id_k \rangle;
    \end{enumerate}
  \item \textbf{end}
  \item \textbf{end}
  \item return \textit{node}(p);
  \end{enumerate}
\end{enumerate}
Algorithm 2: \text{constructArgGraph}(G, q, T, \Sigma, \Delta): Construct trust extended argumentation graph

\textbf{Input:} G: a shared graph structure; q: query; T: trust network; \Sigma: knowledge base; \Delta: rule base

\begin{itemize}
  \item[] G := \emptyset;
  \item[] qNode \leftarrow \text{constructProofNet}(G, q, T, \Sigma, \Delta);
  \item[] if qNode \neq \emptyset then
    \begin{itemize}
      \item[] if qNode is not visited by argumentation construction yet then
        \begin{itemize}
          \item[] defeater \leftarrow \text{constructArgGraph}(G, \neg n, T, \Sigma, \Delta);
          \item[] if defeater \neq \emptyset then
            \begin{itemize}
              \item[] Add a rebut-defeat link \langle defeater, n \rangle to G if it doesn’t exist;
              \item[] Initialize ID_{\text{defeat}}(\langle defeater, n \rangle) if it is not initialized;
              \item[] for each combination \langle id, id' \rangle \in ID(defeater) \times ID(qNode) do
                \begin{itemize}
                  \item[] Add \langle id, id' \rangle into ID_{\text{defeat}}(\langle defeater, n \rangle);
                \end{itemize}
            \end{itemize}
          \end{itemize}
        \end{itemize}
      \end{itemize}
    \end{itemize}
  \item[] for each node n in G connecting to q through reasoning links do
    \begin{itemize}
      \item[] if n \in \Sigma is premise node and it is not visited yet then
        \begin{itemize}
          \item[] defeater \leftarrow \text{constructArgGraph}(G, \neg n, T, \Sigma, \Delta);
          \item[] if defeater \neq \emptyset then
            \begin{itemize}
              \item[] Add a premise-undercut link \langle defeater, n \rangle to G if it doesn’t exist;
              \item[] Initialize ID_{\text{defeat}}(\langle defeater, n \rangle) if it is not initialized;
              \item[] for each combination \langle id, id' \rangle \in ID(defeater) \times ID(qNode) do
                \begin{itemize}
                  \item[] Add \langle id, id' \rangle into ID_{\text{defeat}}(\langle defeater, n \rangle);
                \end{itemize}
            \end{itemize}
          \end{itemize}
        \end{itemize}
      \end{itemize}
    \end{itemize}
  \item[] if n is an intermediate conclusions and it is not visited yet then
    \begin{itemize}
      \item[] defeater \leftarrow \text{constructArgGraph}(G, \neg n, T, \Sigma, \Delta);
      \item[] if defeater \neq \emptyset then
        \begin{itemize}
          \item[] Add a intermediate-undercut link \langle defeater, n \rangle to G if it doesn’t exist;
          \item[] Initialize ID_{\text{defeat}}(\langle defeater, n \rangle) if it is not initialized;
          \item[] for each combination \langle id, id' \rangle \in ID(defeater) \times ID(qNode) do
            \begin{itemize}
              \item[] Add \langle id, id' \rangle into ID_{\text{defeat}}(\langle defeater, n \rangle);
            \end{itemize}
        \end{itemize}
      \end{itemize}
    \end{itemize}
  \item[] if n \in \Delta is an inference rule node and it is not visited yet then
    \begin{itemize}
      \item[] defeater \leftarrow \text{constructArgGraph}(G, \neg (\bigwedge_{p \in \text{Premises}(r)} \{p_i \rightarrow \text{conclusion}(c)\}), T, \Sigma, \Delta);
      \item[] if defeater \neq \emptyset then
        \begin{itemize}
          \item[] Add an inference-undercut link \langle defeater, n \rangle to G if it doesn’t exist;
          \item[] Initialize ID_{\text{defeat}}(\langle defeater, n \rangle) if it is not initialized;
          \item[] for each combination \langle id, id' \rangle \in ID(defeater) \times ID(qNode) do
            \begin{itemize}
              \item[] Add \langle id, id' \rangle into ID_{\text{defeat}}(\langle defeater, n \rangle);
            \end{itemize}
        \end{itemize}
      \end{itemize}
    \end{itemize}
  \end{itemize}
return qNode;
Figure 8: A compact trust-extended argumentation graph
8 Trust propagation

Having shown how to construct trust-extended argumentation graphs, in this section we turn to the question of how to propagate information about trust in those graphs — after all we introduced these graphs exactly to be able to combine trust data with argumentation. Before we can introduce the trust propagation algorithms, we first define the data structure \( \text{trustBel} \) that we use to capture an agent \( Ag_i \)’s trust or belief in a node \( n \):

\[
\text{trustBel}[Ag_i, n] = \begin{cases} 
\text{tr}(Ag_i, n) & \text{if } n \text{ is an agent,} \\
\text{bel}_i(n) & \text{if } n \in \Sigma_i \text{ or } n \in \Delta_i.
\end{cases}
\]

To compute the trust-based belief of an agent over an argument, we employ a set of algorithms: Algorithm 3, Algorithm 4, Algorithm 5 (or Algorithm 6 if we take into account the reasoning structure of the argument when computing the beliefs), Algorithm 8 and Algorithm 9 (computing trust in a depth-first manner) or Algorithm 10 (if we need to compute the trust in a breadth-first manner), with two customized functions combine and fuse.

Algorithm 3 initializes the data structure trustBel, which will hold an agent’s trust in another agent or an agent’s beliefs over premises and inference rules, to NIL. Algorithm 4 computes the trust of one agent in the other agents in a depth-first manner using the customized combine and fuse functions as discussed below. If the trust model requires a breadth-first trust computation (as TidalTrust does), we can then employ Algorithm 10. With the trust computed, we can then use Algorithm 5 to compute beliefs over arguments as a whole. An alternative way to compute beliefs over arguments is to use Algorithm 6 and Algorithm 7 if the underlying belief propagation model requires that we take into account the structure of the argument rather than just the set of premises. Finally, Algorithm 4 invokes the computation of all the agents’ trust and beliefs over the other agents and on the knowledge, inference rules, and the conclusions that can be derived from them.

To deal with trust propagation, a typical implementation for combine is:

\[
\text{combine}\left(\{nd_1, nd_2, ..., nd_n\}\right) = nd_1 \otimes^{tr} nd_2 \otimes^{tr} ... \otimes^{tr} nd_n.
\]

A typical implementation for fuse is:

\[
\text{fuse}\left(\{nd_1, nd_2, ..., nd_n\}\right) = nd_1 \oplus^{tr} nd_2 \oplus^{tr} ... \oplus^{tr} nd_n.
\]

Other implementations of combine and fuse can be constructed depending on the underlying trust model or the belief model being used. For example, in TidalTrust, we will have a breadth-first style implementation and use Algorithm 10 for trust computation:

\[
\text{fuse}\left(\{\text{tr}(Ag_i, Ag_k) | (Ag_i, Ag_k) \in \mathcal{T}\} \cup \{\text{tr}(Ag_k, ag_j) | (Ag_i, Ag_k) \in \mathcal{T}\}\right) = \frac{\sum_{(Ag_i, Ag_k) \in \mathcal{T}} \text{tr}(Ag_i, Ag_k) \text{tr}(Ag_k, Ag_j)}{\sum_{(Ag_i, Ag_k) \in \mathcal{T}} \text{tr}(Ag_i, Ag_k) > \theta \text{tr}(Ag_i, Ag_k)}.
\]

For belief propagation, a typical implementation for combine is:

\[
\text{combine}\left(\{nd_1, nd_2, ..., nd_n\}\right) = nd_1 \otimes^{bel} nd_2 \otimes^{bel} ... \otimes^{bel} nd_n.
\]

and a typical implementation for fuse is:

\[
\text{fuse}\left(\{nd_1, nd_2, ..., nd_n\}\right) = nd_1 \oplus^{bel} nd_2 \oplus^{bel} ... \oplus^{bel} nd_n.
\]

Other implementations of combine and fuse can be constructed depending on the underlying model of belief.

**Proposition 10** Let \( W \) be the number of arguments in the argumentation graph \( G \). Algorithm 4 trustBelPropagate computes the beliefs of an agent \( Ag_i \) on all arguments in the size of \( G \) and the number of arguments, namely \( O(W \cdot |G|) \).

**Proof:** With the help of data structure trustBel[Ag_i, n], we have a dynamic programming-style computation of trust and belief. The algorithm just needs to investigate each edge in \( G \) once, and compute the relevant trust or belief on
Once the value has been computed, it can be looked up in the data structure \( \text{trustBel} \). If we implement this with an appropriate hash table, we can guarantee the look-up for the computed trust and belief value in \( O(1) \), resulting in a total complexity in terms of \( O(W \cdot |G|) \).  

As \( G \) is polynomial in the size of the set of premises and inference rules, we can compute the trust in all arguments with this complexity. As before, the number of arguments can still be exponential in the number of premises and rules, but the way we implement the trust and belief computation through the compact argumentation graph suggests various possibilities to reduce the complexity in our future work. Note that with Algorithm 4, we compute trust values and belief values of the nodes in \( G \) with complexity in the size of \( G \). In this way, we may be able to avoid constructing undefendable arguments in terms of the belief levels of the components of arguments. This is another line of future research.

**Algorithm 3**: \( \text{initTrustMatrix}(G) \): Initialize the trust matrix

<table>
<thead>
<tr>
<th>Input: ( G ): a shared graph structure; ( \mathcal{T} ): trust network; ( \Sigma ): knowledge base; ( \Delta ): rule base</th>
</tr>
</thead>
<tbody>
<tr>
<td>for each agent ( A_g ) do</td>
</tr>
<tr>
<td>for each argument ( \text{arg} \in G ) do</td>
</tr>
<tr>
<td>( \text{trustBel}[\text{Ag}_i, \text{arg}] \leftarrow \text{NIL} );</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>for each premise and inference node ( \text{nd} \in G ) do</td>
</tr>
<tr>
<td>( \text{trustBel}[\text{Ag}_i, \text{nd}] \leftarrow \text{NIL} );</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

**Algorithm 4**: \( \text{trustBelPropagate}(G, \text{computeBel}, \text{combine}, \text{fuse}) \): Trust and belief propagation in argumentation

<table>
<thead>
<tr>
<th>Input: ( G ): a shared graph structure; ( \text{computeBel} ): a customized function to compute trust and belief; ( \text{combine} ): a customized function to combine (AND-input) multiple sources of trust and belief together; ( \text{fuse} ): a customized function to fuse (OR-input) multiple sources of trust and belief together</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{InitTrustMatrix}(G) );</td>
</tr>
<tr>
<td>for each agent ( A_g ) do</td>
</tr>
<tr>
<td>for each argument ( \text{arg} \in G ) do</td>
</tr>
<tr>
<td>( \text{trustBel}[\text{Ag}_i, \text{arg}] \leftarrow \text{computeBel}(G, \text{Ag}_i, \text{arg}, \text{combine}, \text{fuse}); )</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

**Algorithm 5**: \( \text{computeBel}(G, \text{Ag}_i, \text{arg}, \text{combine}, \text{fuse}) \): Compute the trust or belief of \( A_g \) over node in argument \( \text{arg} \)

<table>
<thead>
<tr>
<th>Input: ( G ): a shared graph structure; ( \text{Ag}_i ): an agent; ( \text{combine} ): a customized function to combine (AND-input) multiple sources of trust and belief together; ( \text{fuse} ): a customized function to fuse (OR-input) multiple sources of trust and belief together</th>
</tr>
</thead>
<tbody>
<tr>
<td>for each node ( \text{nd} \in \text{arg} ) do</td>
</tr>
<tr>
<td>( \text{bel}[\text{nd}] \leftarrow \text{computeTrustBel}(G, \text{Ag}_i, \text{nd}, \text{combine}, \text{fuse}); )</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>( \text{trustBel}[\text{Ag}_i, \text{arg}] \leftarrow \text{combine}({ \text{bel}[\text{nd}]</td>
</tr>
<tr>
<td>return ( \text{trustBel}[\text{Ag}_i, \text{arg}] );</td>
</tr>
</tbody>
</table>
Algorithm 6: \( \text{computeBel}(G, Ag_i, arg, combine, fuse) \): Compute the trust or belief of \( Ag_i \) over node in argument \( arg \) taking into account the argument structure

**Input:** \( G \): a shared graph structure; \( Ag_i \): an agent; \( combine \): a customized function to combine (AND-input) multiple sources of trust and belief together; \( fuse \): a customized function to fuse (OR-input) multiple sources of trust and belief together

\[
\text{trustBel}[Ag_i, arg] \leftarrow \text{computeTrustBel}(G, Ag_i, arg, \text{conclusion}(arg), combine, fuse);
\]

**return** \( \text{trustBel}[Ag_i, arg] \);

Algorithm 7: \( \text{computeTrustBel}(G, Ag_i, arg, node, combine, fuse) \): Compute the trust or belief of \( Ag_i \) over node in argument \( arg \) taking into account the argument structure

**Input:** \( G \): a shared graph structure; \( Ag_i \): an agent; \( combine \): a customized function to combine (AND-input) multiple sources of trust and belief together; \( fuse \): a customized function to fuse (OR-input) multiple sources of trust and belief together

if node is a premise node or inference node then
    return \( \text{computeTrustBel}(G, Ag_i, node, combine, fuse) \);
end

if node is a conclusion node then
    \( \gamma \leftarrow \text{inference rule connecting to node in arg} \);
    \( \text{result}[\gamma] \leftarrow \text{computeTrustBel}(G, Ag_i, \gamma, \text{combine}, fuse) \);
    for each premise \( pm_k \) connecting to \( \gamma \) in arg do
        \( \text{result}[pm_k] \leftarrow \text{computeBel}(G, Ag_i, arg, pm_k, combine, fuse) \);
    end
    return combine(\( \{\text{result}[pm_k]\} \cup \{\text{result}[\gamma]\} \));
end

Algorithm 8: \( \text{computeTrustBel}(G, Ag_i, node, combine, fuse) \): Compute the trust or belief of \( Ag_i \) over node in argument \( arg \)

**Input:** \( G \): a shared graph structure; \( Ag_i \): an agent; \( combine \): a customized function to combine (AND-input) multiple sources of trust and belief together; \( fuse \): a customized function to fuse (OR-input) multiple sources of trust and belief together

if \( Ag_i \) holds node then
    return \( \text{Bel}_i(node) \);
end

if \( \text{trustBel}[Ag_i, node] \neq \text{NIL} \) then
    return \( \text{trustBel}[Ag_i, node] \);
end

for each agent \( Ag_j \) who holds node do
    \( \text{res} \leftarrow \text{computeTrust}(Ag_i, Ag_j) \);
    if \( \text{res} \neq \text{NIL} \) then
        \( \text{bel}[Ag_j] \leftarrow \text{combine}(\text{res}, \text{Bel}_j(node)) \);
    end
end

\( \text{trustBel}[Ag_i, node] \leftarrow \text{fuse}(\{\text{bel}[Ag_j]|Ag_j \text{ holds node}\}) \);

**return** \( \text{trustBel}[Ag_i, node] \);
Algorithm 9: computeTrust(G, Agi, Agj, combine, fuse): Compute the trust or belief of Agi over node in argument arg in a Depth-First manner

Input: G: a shared graph structure; Agi: an agent; combine: a customized function to combine (AND-input) multiple sources of trust and belief together; fuse: a customized function to fuse (OR-input) multiple sources of trust and belief together

if trust[Agi, Agj] ≠ NIL then
    return trust[Agi, Agj];
else if (agj, Agi) ∈ T then
    return tr(agj, Agi);
else if there is no Agk such that ⟨Agi, Agk⟩ ∈ T then
    return NIL;
else
    for each Agk such that ⟨Agi, Agk⟩ ∈ T do
        res ← computeTrust(Agk, Agi);
        if res ≠ NIL then
            result[agk] ← combine(tr(agk, Agi), res);
        end
    end
end

Algorithm 10: computeTrust(G, Agi, Agj, combine, fuse): Compute the trust or belief of Agi over node in argument arg in a Breadth-first manner

Input: G: a shared graph structure; Agi: an agent; combine: a customized function to combine (AND-input) multiple sources of trust and belief together; fuse: a customized function to fuse (OR-input) multiple sources of trust and belief together

if trust[Agi, Agj] ≠ NIL then
    return trust[Agi, Agj];
else if (agj, Agi) ∈ T then
    return tr(agj, Agi);
else if there is no Agk such that ⟨Agi, Agk⟩ ∈ T then
    return NIL;
else
    for each Agk such that ⟨Agi, Agk⟩ ∈ T do
        result[agk] ← computeTrust(Agk, Agi);
    end
    trust[Agi, Agj] ← fuse(tr(Agi, Agk)|⟨Agi, Agk⟩ ∈ T) ∪ {result[agk]|⟨Agi, Agk⟩ ∈ T};
end
9 Discussion

As we mentioned in the introduction, there is some consensus in the trust literature that provenance, the relating of information to its source, plays an important role in reasoning about trust. This connection is made explicitly in [30] and [32], but can be found implicitly in, for example, work like [51] which identifies internet authorities by their links with other nodes, and [2] which constructs reputations for individuals from their contributions to online media. Since argumentation explicitly captures the data used in reasoning, it seems a natural tool for dealing with provenance, and the work we describe here develops a system of argumentation that not only records what data is used but also ties this data back to the individuals who provided it.

The provision of such a model — a graphical model which goes beyond existing graphical accounts of Dung-like argumentation — is one part of the contribution of this work. The other part is the provision of algorithms for constructing the graphical model, which identifies the arguments, and then propagates trust and belief values through the resulting network, quantifying the conclusions of the arguments. Though this particular combination of argumentation and trust is novel, the idea of combining trust and argumentation is not. Four lines of work on trust and argumentation that are complementary to ours include those of Harwood [38, 39], Matt at al., Hunter [40], [58], and Stranders [84]. In addition, argumentation has been used in the past to reason about risk [47, 60], a subject closely related to trust though the cited work looked at risk of carcinogenicity given chemical structure rather than risk due to untrustworthiness.

Harwood [38, 39] takes a strong position against the use of numerical estimates of trust, instead constructing an elegant and purely symbolic approach. In his work, arguments are constructed about the reasonableness or otherwise of trusting individuals. Given such a set of arguments, it is then possible to impose a semantics reminiscent of Dung’s which identifies not which arguments are acceptable, but which are to be trusted and distrusted. Hunter [40], meanwhile, is not interested in explicitly modelling trust, but is concerned with reasoning about the proponents of arguments. He does this through meta-level reasoning about arguments, that is, he constructs arguments about how good arguments are, where the grounds of these meta-arguments include information about the agents who put forward the object-level arguments.

Both Harwood and Hunter take strictly symbolic approaches that are not concerned with numerical degrees of trust. In contrast, Matt at al. [58] are, like us, interested in combining numerical data on trustworthiness with arguments. However, where we take the trust ratings for granted — or, alternatively, assume that we will employ well-founded means of establishing trust ratings from data — Matt at al. describe how such trust ratings may be constructed. The closest of these four lines of work to ours is the work of Stranders [84] who use the argumentation system from [5] to handle trust. This is a less general approach than ours, being tied to a possibilistic notion of trust, but they extend [5] to include vague propositions. These are propositions that, unlike those we deal with, are not just true or false, but have fuzzy degrees of truth, giving the system greater representational power.

All of the systems we have just mentioned suggest future directions for our work, and we have already mentioned areas of future work in connection with the computation of arguments and the propagation of values. Another important area of future work is to establish how to use the trust values we compute within argumentation. In Section 6, we discussed using trust values to provide strengths for arguments, following [4] in modifying defeat so that the defector has to be stronger (in this case more trusted) than the defeatee. There are other ways to use trust values. One natural use for trust values is to allow agents to specify a trust threshold — all information from agents trusted less than the threshold is discarded. A less obvious use, but one we find intriguing, is the use of a trust budget (akin to the inconsistency budget of [24]). This approach would allow an agent to decide on a maximum amount of distrust it was prepared to tolerate across all the agents who provided it with information pertaining to a single argument, and obtain different sets of conclusions for different budget levels. Such an approach would allow the agent to search for particular conclusions it was interested in (plans to achieve particular goals for example) and determine exactly how trusting it had to be to get the conclusions it wanted.

Finally, we need to acknowledge that both the model of trust that we have used, and the model of agent beliefs, are very simple. Trust is represented as a simple numerical measure, in most cases a single number, and a 3-tuple in the case of subjective logic. An agent’s beliefs are modelled by attaching similar numerical measures to formulae. While most work on argumentation takes a similarly simple view of belief, this does not mean that argumentation can only work with such a simple model. As an example, in [71] we described a system of argumentation that used a multi-model logic, in particular the belief/desire/intention logic of Rao and Georgeff [74, 75]. The model makes use
of multi-context systems [31] to capture the different modalities, and we can use the same techniques to capture the modal notions of trust and belief from [15, 57].

10 Conclusions

In this paper, we have introduced a general approach to combining argumentation with information about trust in a way that allows us to consider information about the degree to which other agents are trusted when reasoning with information obtained from them. The approach we introduce makes no commitment to a specific mathematical approach to computing trust — it can be instantiated with any of a number of numerical systems for propagating trust information that have been proposed in the literature. In particular, we discussed how to use possibility values, as suggested by [79], subjective logic, as suggested by [90], and the TidalTrust approach [50]. In addition to introducing this system, we explored a number of the basic properties of the system, and illustrated its use on an example from [50], showing how our system obtains the same solution as [50]. We also discussed in detail some of the computation required by our approach. In particular, we described how to construct arguments, and how to propagate trust and belief values through the graphical representation of arguments that we use.

Future directions, as mentioned above, include detailed examination and comparison of different approaches to propagating and combining qualitative and quantitative values for trust and belief. Long term work concerns development of an interactive tool to support users making the types of decision faced by John in our extended example.

Acknowledgement

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

We would like to thanks the reviewers. Their many helpful suggestions have greatly improved this paper.

References


[26] http://trust.mindswap.org/FilmTrust/


