Trust and Matching Algorithms for Selecting Suitable Agents

NARDINE OSMAN and CARLES SIERRA, Artificial Intelligence Research Institute (IIIA-CSIC)
FIONA MCNEILL, School of Informatics, the University of Edinburgh
JUAN PANE, Department of Information Engineering and Computer Science, University of Trento
JOHN DEBENHAM, Centre for Quantum Computation & Intelligent Systems, University of Technology Sydney

This paper addresses the problem of finding suitable agents to collaborate with for a given interaction in distributed open systems, such as multiagent and P2P systems. The agent in question is given the chance to describe its confidence in its own capabilities. However, since agents may be malicious, misinformed, suffer from miscommunication, and so on, one also needs to calculate how much trusted is that agent. This paper proposes a novel trust model that calculates the expectation about an agent's future performance in a given context by assessing both the agent's willingness and capability through the semantic comparison of the current context in question with the agent's performance in past similar experiences. The proposed mechanism for assessing trust may be applied to any real world application where past commitments are recorded and observations are made that assess these commitments, and the model can then calculate one's trust in another with respect to a future commitment by assessing the other's past performance.

Categories and Subject Descriptors: I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms: Algorithms

Additional Key Words and Phrases: semantic matching, trust and reputation

ACM Reference Format:
Osman, N., Sierra, C., McNeill, F., Pane, J., and Debenham, J. 2011. Trust & Matching Algorithms for Selecting Suitable Agents. ACM Trans. Intell. Syst. Technol. V, N, Article A (January YYYY), 40 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

The ability of automated agents to interact online creates enormous potential for fast and effective information sharing and service provision, without the need for intensive human intervention. However, it also raises many difficulties. One such difficulty is how to determine whether or not to trust other agents, who may be unable to do the things they say they can, or may be intentionally dishonest, or may provide information or services of a poor quality. Another such difficulty is how agents can communicate even when their data sources have been developed independently and may therefore be very different: how are they to determine exactly what is required of them in an interaction, and how can they tell if they are able to provide this?

Author's addresses: N. Osman and C. Sierra, Artificial Intelligence Research Institute (IIIA-CSIC); F. McNeill, School of Informatics, University of Edinburgh; J. Pane, Department of Information Engineering and Computer Science, University of Trento; J. Debenham, Centre for Quantum Computation & Intelligent Systems, University of Technology, Sydney.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

When an agent wishes[1] to perform an action or discover some information, it must find other agents that are willing and able to provide this information or service, and it must find some way to determine how this interaction will proceed. For example, an agent wishing to buy a ticket must find a ticket-selling agent, and there must be some way for both agents to understand what they are to do in this interaction.

The latter problem is most often solved by engineering models of interactions, and examples of these approaches are electronic institutions [Arcos et al. 2005] and distributed dialogues [Robertson 2005]. However, this paper does not address how an interaction is specified or agreed upon (which may require some argumentation). It assumes an interaction model exists, and tries to find suitable collaborators for it.

In other words, it is the former problem that this paper chiefly addresses.[2] We believe that there are two attributes that are particularly suggestive of how well an agent will be able to perform. The first attribute needs to measure how confident the agent in question is about its capability to play a given role. In other words, do its capabilities *match* what it is required to do? However, answering this question alone is not enough. This paper focuses on open distributed systems, where the processes and infrastructure of these agents will not be opaque and their good intentions (or willingness) are not guaranteed. Any claims these agents make cannot be taken at face value. Agents may be intentionally fraudulent; may believe they can perform tasks which it later transpires they cannot perform, or can only perform poorly; may suffer from confused communication with other agents they interact with and may thus perform their task in an unexpected manner or with unexpected and undesired outcomes; may be poorly connected to the network and may therefore vanish in the middle of an interaction; and so on. As such, there is a need for a second attribute, which answers the question of whether the agent may be expected to generally perform well in a given role. In other words, can it be *trusted* to play that role?

The contributions of this paper are three-fold:

— We introduce our **trust algorithm**, which aids in selecting collaborators by calculating trust measures that measure the expected performance of agents by analysing the agents' performance in 'similar' past experiences. The assessment is made for a given context, where the given context is defined by the chosen interaction model. For instance, if a seller has a good reputation in selling beer, then it will be expected to have a higher probability of performing better in selling wine than selling fish.
The proposed trust model follows both socio-cognitive and experience-based approaches. In other words, it relies on past experiences to predict whether the agent is both capable and willing to perform the action it is set to execute. An interesting aspect of this algorithm is that it provides a generic mechanism for assessing trust that may be applied to any real world application where past commitments are recorded and observations are made that assess these commitments.
We note that the proposed model considers only the degree of capability and willingness of an agent. They do not consider other aspects, such as feelings. Nevertheless, although we focus on artificial agents in multiagent systems, the theory may still be used for human agents too as long as the roles and responsibilities expected from a human in a given interaction are clearly defined and past performance is recorded. For example, the same approach may be used in eBay (if past experiences

[1] We refer to an agent's wishes and beliefs, although these are in fact the wishes of its designer and the facts in its knowledge base.

[2] Although since this is happening in a particular context, we must make decisions about how these interactions are proceeding, which are explained in Section 2. The majority of the work described in this paper would be wholly or partly applicable to different ways of determining interactions, but our implementation and evaluation are based on the context we describe in Section 2.

were recorded with greater accuracy) to figure out which seller to trust most when considering the quality of goods, quality versus price ratio, promptness in shipment, quality of customer service, etc.

— We discuss our **matching algorithm**, which allows an agent to assess how well its own data source matches the capabilities required by a particular role it may wish to take on. This describes the agent's self confidence in how well it can play a given role. This algorithm matches first-order terms, so is applicable to interactions in which the abilities required of an agent can be represented as first-order terms, and agents which have structured data which can also be represented as first-order terms. This is fairly broadly applicable: for example, we have implemented a translation process to represent WSDL service interfaces appropriately, and it would be straightforward to do this for database entries. Our algorithm returns a score indicating the quality of the match.
Note that the extensive details of this algorithm's implementation and evaluation have been published elsewhere (full details can be found in [Giunchiglia et al. 2008]). We include a section on it in this paper because it is important to place it in the context of finding the best agent to interact, considering both trust and matching. However, we go into much less detail than we do for the trust algorithm, and point the interested reader to relevant papers.

— We present our **good enough answers algorithms**, which combines trust and matching to provide an estimate of how well an agent will perform in a given situation. The two scores calculated above are orthogonal, and yet both are important when determining how well an agent will perform. Determining how such a combination should be done to provide the best results is difficult and context-dependent, and we believe that only very extensive testing across a broad range of environments and requirements can produce a definitive answer. However, we present two algorithms which take different approaches to combining these scores, and justify why we believe these are likely to produce good estimates of an agent's overall ability.
We believe that this 'holistic' approach (as opposed to focusing only on trust, or only on data matching) is unique, and that the ability to consider both aspects, and how they may be combined, is essential to informed agent selection.

The rest of this paper is divided as follows. Section 2 opens with a motivating example. Section 3 presents our proposed trust model, Section 4 presents a matching algorithm that is complementary to the trust one, and Section 5 illustrates how the trust and matching scores may be combined into one final performance measure. Section 6 provides a final discussion on the proposed models and Section 7 presents a summary of results. Finally, Section 8 provides a background on the related work in this field, before concluding with Section 9.

2. MOTIVATING EXAMPLE

An area where access to fast, reliable information is essential is that of disaster response. In such situations, there may be some information that is highly safety critical, in which human intervention (for example, permission for an action being granted by someone high up the chain of command) is necessary. However, sharing information only via humans is enormously limiting, and depends on the number of people available to participate, the possibility of contacting these people, and the awareness of these people of what data is available (possibly from very large data sources). Many investigations into the response to disaster events (such as the Pitt Report [Pitt 2007] into UK flooding in 2007) highlight the failure to effectively share information in a speedy and appropriate manner as a major hinderance to the success of the response.

In such situations, there are many difficulties with automated information sharing, of which we are particularly interested in two:

— Information from different organisations, or even from different branches of the same organisation, will be organised differently, both semantically (that is, different terms will be used for the same or similar things) and structurally (for example, fields in a database record may be in a different order, or one database may have more fields than another). In order for this information to be mutually comprehensible, $matching$ is required.
— There may be many responders involved, from large organisations (such as governments and police departments) to local business and charities and even individuals. The quality of information that these organisations have will not be equal. Some may not be in a position to have such accurate or up-to-date information as others; some may even be deceitful or unreliable. Determining the quality of information therefore relies on considering the $trust$ which is placed in the information-providing organisation. This is not simply a matter of ranking organisations according to authority, but is context-dependent. For example, the police department may be the authority of which roads have been closed, but though they will also have access to weather information, this is likely to be at least second-hand. An individual on the street will in general not be considered particularly authoritative, but may have excellent information on an event which is unfolding right in front of them. It is therefore important to incorporate a context-dependent notion of trust when determining how much reliance to place on information.

The combination of these two factors leads us to our notion of *good enough* answers: if I pose a particular question, how can I determine the value to place on the answer received from a particular organisation. This depends both on the matching and the trust score. From the matching point of view, we must consider questions such as how similar is the response to the question I posed, and were any elements of my question unmatched? For example, if I am searching for information about the whereabouts of vulnerable people, posed as *location(Person,Coordinates,Vulnerability)*, then the response *whereabouts(Human,Coordinates,Vulnerability)* may be considered a better match than the response *location(Person,Coordinates)* even though it uses slightly different words, because the second response left out an attribute that may be considered vital. From the trust point of view, we must consider whether the information provider is appropriate: for example, the local hospital would be considered more trustworthy on this particular subject than an unknown individual. But if the best match is from the less trusted authority, should we prefer this to a poorer match from a reliable authority? This question is very difficult to answer in the general case, but is the central question we wish to answer in this paper. We present two automated solutions to this puzzle in Section 5.

We believe that this approach is general purpose, and can be adapted to many different situations. However, our current implementation, and the evaluation of this which we discuss in Section 7, are within the OpenKnowledge project[3]. We briefly introduce some central concepts of this project, in order to clarify our approach.

The Open Knowledge project focused on facilitating interactions between agents (or services, systems, etc.) through the use of explicit, shared interaction models (IMs), which provide information about what roles must be played in the interactions, what messages should be passed at each stage, and what actions are required of players. These IMs are written in the Lightweight Coordination Calculus (LCC) [Robertson 2005]. Figure 1 presents an example IM. The different agents, or *roles*, are indicated

[3]www.openk.org

in the term *a(Role_Name,Player)*, and each must have a set of messages they must send and receive, and the constraints they must satisfy to be able to pass the messages. Message passing is indicated by a double arrow, whereas the constraints that must be satisfied in order to allow a particular message to be sent are indicated by a single arrow. An IM must provide this information for every role, and they must be compatible across the different roles: in our example, the player of the *police* role must send a message *water_level(Location)* to the player of the *sensor* role; therefore the *sensor* role must expect this message from the *police* role.

The *meaning* of an LCC interaction is encoded in the constraints. The purpose of the messages is to share information about the instantiation of variables, and to make it explicit what stage the interaction is at. It is possible, for example, to relabel *water_level(Location)* as *message1(Location)*, and informative names are chosen only to aid human readability. The naming of constraints, however, is crucial, and encodes information about what the constraint means. For example, in order to send the first message, the police agent must satisfy the constraint *suitable_loc(Entity,Location)*. If the police agent has been designed with this particular interaction in mind, or if the IM was designed by the same person who designed the police agent, it is likely that this exactly reflects the organisation of data in the police agent's data source, and it will be able to unify this constraint with a fact in its database and return the answer. However, in an open environment, where IMs are often reused by different parties, it will often happen that the police agent does have information about suitable locations, but does not represent it in exactly this fashion. Our matching techniques (see Section 4) allow the police agent to match its own representation to that of the constraint, so that it can still satisfy the constraint. This match will often not be perfect (if the information content of the two terms is similar but not identical), so the agent's ability to satisfy a constraint will not be perfect. If an agent cannot satisfy a constraint either through unification or matching, then it is unable to perform the role. Once constraints have been satisfied, messages are passed automatically.

$a(police, P)$ ::
$water\_level(Location) \Rightarrow a(sensor, S) \leftarrow drop\_off(Entity) \wedge suitable\_loc(Entity, Location)$ then
$water\_level(Location, Level) \Leftarrow a(sensor, S)$ then
$drop\_off(Entity, Location) \Rightarrow a(firefighter, F) \leftarrow safe\_level(Level).$

$a(sensor, S)$ ::
$water\_level(Location) \Leftarrow a(police, P)$ then
$water\_level(Location, Level) \Rightarrow a(police, P) \leftarrow detect(Location, Level).$

$a(firefighter, F)$ ::
$drop\_off(Entity, Location) \leftarrow drop\_off(Entity, Location) \Leftarrow a(police, P).$

Fig. 1. Sample interaction model scpecified in LCC

The first step in enacting an IM is to determine which agent should take on which role. Agents must subscribe to roles which they wish (and believe they are able) to play. Before interaction, they can look at which agents are signed up to other roles and decide which of these agents, if any, they wish to interact with. Once all agents have made these decisions, a *negotiation* agent will assign roles in a way that respects these issues.

Determining how one can decide which other agents it is most advantageous to interact with is the question that this paper addresses. For example, in the above IM,

consider an organisation with an agent which intends to take on the role *police*. Playing this role will involve interacting with another agent (or service) playing the role *firefighter*, as well as agent or agents playing the role *sensor*. Although these sensors are likely to be automated, the trust and matching issues remain the same, and the interaction will only produce a satisfactory outcome if that other agent plays its role to an acceptable standard. If there are many agents vying for this role then it is useful to be able to rank them according to which is likely to perform the role best; even if only one agent is interested in playing a role, it is still desirable to check that the performance of this agent is likely to meet a minimum required standard. The degree to which an agent can play a role depends on the degree to which it can satisfy all of the constraints on that role, since message passing is automatic once the relevant constraint has been satisfied. To judge whether or not an agent can satisfy a constraint well, we need to consider the following:

— How good do we think this agent is at performing this task? For example, if they are providing information about the whereabouts of people, is their information about people's locations usually of high quality? This score is determined using our *trust* algorithm (Section 3).
— How well can the information contained in the agent's ontology (or knowledge source) match the constraint? For example, is the agent able to fulfil all aspects of the constraint? This score is determined through our *matching* algorithm (Section 4).[4]

These scores are combined using our *good enough matching* algorithm (Section 5), and then by combining these scores for all pertinent constraints we are able to produce an automated estimate of which agent is likely to perform best in the given role.

3. THE TRUST MODEL

The proposed trust model follows the most basic definition of socio-cognitive approaches. In such a model, one agent's belief about the other's capability to perform a given action and its willingness and persistence to actually carry out that action is crucial in determining whether the latter agent is to be trusted or not by the former [Castelfranchi and Falcone 1998; 2000]. Furthermore, our model calculates the belief about another's *capability* and *willingness* as probability measures. In addition to socio-cognitive approaches, the calculation follows the experience-based trust models. In other words, to calculate one agent's belief about another's capabilities and willingness in performing various actions depends on observing and learning from its *past performance* [Schillo et al. 2000; Abdul-Rahman and Hailes 2000; Sabater and Sierra 2002]. With this foundational background, we introduce our view on trust and the resulting trust model.

Our basic tenet on trust is that it may be defined as a cognitive state that an agent α holds with respect to the expected behaviour of another agent β on some matter φ. We assume that each agent has a fixed[5] local ontology O and that φ is a correct term

[4]This score is calculated by the agent which intends to perform the role. The scores are made public but they are not possible for other agents to replicate because we assume that the knowledge sources of external agents are private, and therefore the information used to obtain the score is not available. This raises the possibility that agents may lie about their scores and present themselves as much better at performing the role than they really are. Whilst we accept that this may happen, we believe that this will be balanced out by the trust score. If an agent takes on a role which it performs badly, its trust score will be greatly lowered, so that on subsequent interactions, even if it presents a high matching score, its low trust score will mean that its combined score is not very high. It is therefore not possible to game the system long-term in this way.

[5]Ontologies certainly evolve; however, the process by which an ontology evolves is out of the scope of this paper. We assume though that it does not change *within* a particular interaction.

from the set of terms built on top of O, denoted as $Term(O)$.[6] Our view is based on a relation between *commitment*, what β promises to do, and *observation*, what α actually observes happening. In probabilistic terms, this could be naturally modelled as a conditional probability: $P(Observing(\alpha, \varphi)|Committed(\beta, \alpha, \varphi))$, that is, the probability of α observing φ given that β made a commitment to α to perform φ. We argue that probabilities are defined to measure expected behaviour. For instance, the probability of picking a heart from a deck of cards is $\frac{13}{52} = 0.25$. However, the probability of picking a second heart from that same deck of cards becomes $\frac{12}{51} = 0.235...$, since the act of picking the second card is now influenced by the previous act of picking the first heart. As such, we also use probabilities to measure the expected behaviour of agents, and we say expected future behaviour is influenced by (or conditional on, in probabilistic terms) past behaviour.

This section is on how to estimate $P(Observing(\alpha, \varphi)|Committed(\beta, \alpha, \varphi))$. As in traditional socio-cognitive trust models, our view is that calculating the probability of observing the action φ that agent β has committed to perform could be based on the capability of agent β to perform φ as well as its willingness to do so. For instance, a firefighter might have the capability of picking up people from dangerous locations and dropping them at safer places, but during a given interaction, the decide not to drop the people at the designated location because it decides that it wants to rescue its loved ones first. In other words, to calculate the probability of observing an agent act, one should calculate the probability of both the agent's capability and its willingness. Consequently, $P(Observing(\alpha, \varphi)|Committed(\beta, \alpha, \varphi))$ is then defined as:

$$
\begin{aligned}
&P(Observing(\alpha, \varphi)|Committed(\beta, \alpha, \varphi)) = \\
&P(Can(\beta, \varphi) \textbf{ and } Does(\beta, \varphi)|Committed(\beta, \alpha, \varphi)) = \\
&P(Can(\beta, \varphi)|Committed(\beta, \alpha, \varphi)) \cdot P(Does(\beta, \varphi)|Committed(\beta, \alpha, \varphi))
\end{aligned}
$$

where $P(Can(\beta, \varphi)$ describes the probability of β having the capability to perform φ, and $P(Does(\beta, \varphi)$ describes the probability of β having the willingness and actually performing φ. Note that the second equality holds because in probability, $P(A \textbf{ and } B) = P(A) \cdot P(B)$ if A and B are independent. This applies to our case because we assume capabilities and willingness to be independent, as we illustrate next.

We assume agents may be capable of performing actions they are not willing to perform and vice versa. In fact, in our work, we do not only consider rational agents that only make commitments that they *know* they can fulfil, but we also consider fraudulent and malicious agents that may lie about their capabilities, or agents that might be mistaken about their own capabilities. In such systems, not only capabilities are unrelated to (and independent from) willingness, but they are also unrelated to commitments. One can commit to an action, regardless of whether or not the action can be performed. A commitment is made based on the *motives* of the agents. For instance, a malicious and greedy agent may commit to actions it cannot carry out in order to fulfil its own goals; a sincere agent that cannot lie or cheat would not commit to actions they cannot perform, although even a sincere agent may commit to an action it thinks it can perform, only to realise later that such a performance is not actually possible. As such, we say one can study whether an agent is capable of performing a given action independently of the commitments it had made. In probabilistic terms, $P(Can(\beta, \varphi)|Committed(\beta, \alpha, \varphi)) = P(Can(\beta, \varphi))$ because the capability is independent of the commitments made. Consequently, $P(Observing(\alpha, \varphi)|Committed(\beta, \alpha, \varphi))$

[6] In some cases, the content of a message can be a Cartesian product of terms instead of a single term. All the ideas of this paper apply naturally to these cases.

is then defined as:

$$P(Observing(\alpha,\varphi)|Committed(\beta,\alpha,\varphi)) = \\ P(Can(\beta,\varphi)) \cdot P(Does(\beta,\varphi)|Committed(\beta,\alpha,\varphi)) \quad (1)$$

We will estimate $P(Does(\beta,\varphi)|Committed(\beta,\alpha,\varphi))$ based on past observed behaviour in similar circumstances (similar commitments). For instance, if the firefighter did follow orders in the past, then the probability that he will be willing to follow orders in the future should be high. As for $P(Can(\beta,\varphi))$, we will estimate it as a matching degree between the capabilities needed now and the capabilities observed in the past. For instance, if the police officer was capable of knowing at some point in the past where each entity (such as students, medics, cattle, etc.) should be transported to in the case of emergency, then the probability of this police officer to still hold this capability should be high (regardless of whether he is willing to act upon this capability or not). As such, estimating a capability is based on deciding whether an observed past capability matches (to a certain degree) the one in question. Then, we shall approximate trust by an entropy measure of the resulting probability distribution $P(Observing(\alpha,\varphi)|Committed(\beta,\alpha,\varphi))$, which we will refer to as $P(\varphi_o \mid \varphi_c)$ for simplification, where φ_o represents the observed action and φ_c the action committed to.

We do note that observing, or not observing, an action cannot always be used as a guarantee on whether the agent did or did not perform the action. For example, a message (such as when sending one's payment) may simply be lost on its way. However, if an agent is known to have a lossy connection, then its computed expected behaviour should rightly take such errors into consideration. For outsiders, this agent cannot be trusted. We believe that focusing on the end results is reasonable when computing the probability of future end results.

In what follows, Section 3.1 provides the preliminaries needed for calculating $P(\varphi_o \mid \varphi_c)$, whose calculation details are then presented by Section 3.2. Section 3.3 presents a set of various methods that may be used for calculating trust scores based on the probability $P(\varphi_o \mid \varphi_c)$. Section 3.4 then provides a summary by presenting a sample algorithm for calculating trust.

3.1. Preliminaries

Before presenting our approach for calculating $P(Observing(\alpha,\varphi)|Committed(\beta,\alpha,\varphi))$, this section provides some preliminaries that our proposal is based on. Our proposal is based on the idea that agents can play different roles in several interactions. For instance, one may play the role of a firefighter in one scenario, and the role of a citizen in another. When selecting the agent to interact with, this agent is selected to play a specific role in a given interaction and this specific role and interaction is crucial for the selection process. For instance, one may be a reputable citizen who is always willing to help, but a terrible firefighter who does not even know how to drive a fire truck. As such, the *context* is crucial. Since our proposed model is an experienced based model that relies on past experiences to predict future performance, calculating the *similarity* between *experiences* is crucial. However, from the point of view of trust (and the probability distribution $P(Observing(\alpha,\varphi)|Committed(\beta,\alpha,\varphi))$), experiences are defined in terms of one agent's *commitments* to perform certain actions, and another agent's *observation* of these actions. As such, the notions of *context*, *commitments*, *observations*, *experiences*, and *similarity measures* are all presented by the following sections, respectively.

3.1.1. Context and Local Ontology. As illustrated earlier, our view is based on having agents play different roles in different interactions. To compare two different contexts,

one should compare the interaction scenario (for instance, an emergency response scenario, an e-commerce scenario, an auction scenario, a Black Jack game scenario, and so on), the role played by the agent (a seller, a buyer, a firefighter, and so on), and the agent's commitments with respect to a given role and interaction (for example, in an e-commerce scenario, the buyer commits to pay for its purchase, the seller commits to deliver on time, and so on).

To be capable of using the proposed trust model of this paper, what is needed is the name of the interaction scenario, the role played by the agent, and the set of actions the agent has committed to in this role. A descriptive text accompanying each may also be used.

Various approaches in multi-agent systems have been proposed for specifying interaction models [Arcos et al. 2005; Robertson 2005]. However, the running example of this document uses the lightweight coordination calculus (LCC), a process calculus, which has been briefly introduced earlier in Section 2. The specification of Figure 1 is an example of a context specified in LCC.

Naturally, each agent has a local ontology that allows it to comprehend the specification of the interaction scenario, the roles of a given interaction, and the individual commitments associated with each role (or agent). Comparing contexts, which is the basis of our proposed model, is based on comparing commitments, roles, and interactions. This is achieved by analysing one's ontology and the matching degree between its terms. For this reason, we first provide a brief introduction to our view of local ontologies.

We assume an agent's ontology O consists of some basic (and finite) atomic terms T with a refinement relation $< \subseteq T \times T$ defining specificity. For example, the refinement relation can specify that a car is a specific example of a $vehicle$. We can naturally extend this relation to tuples of values ($t_1 \times \cdots \times t_n$) as well as to terms built from a function symbol and a set of arguments ($f(t)$), accordingly:

— If $t_i < t'_i$ then $(t_1 \times \cdots \times t_i \times \cdots \times t_n) < (t_1 \times \cdots \times t'_i \times \cdots \times t_n)$
— If $f : T \rightarrow T'$, $g : T \rightarrow T'$, and $g < f$, then $\forall t \in T \bullet g(t) < f(t)$ and $\forall t, t' \in T \bullet t < t' \Rightarrow f(t) < f(t')$

We denote the free algebra of terms generated from O as $Term(O)$. Note that if functions are not recursive then the free algebra is finite. We assume finiteness in the trust computation later on.

The $<$ relation over the set of terms defines a directed acyclic graph. We say, if $t < v$ then t is a descendant of v. The levels of the free algebra define the parent/child relationship. In other words, if $t < v$ and there is no w such that $t < w$ and $w < v$ then v is the father of t. Such graphs can be generated on the fly and locally to a particular term; that is, all ancestors, siblings, and descendants around the term are generated for a given distance from the term, so as to focus on a particular region of the terminology. This on the fly and local generation of graphs is crucial for the efficiency of our trust algorithm, which makes use of these graphs to calculate the similarity between terms based on how far they are in the graph.

3.1.2. Commitments. In our proposed model, evaluating an agent's performance is based on comparing what the agent has committed to do to what it actually did (or what has been observed). Hence, the notion of commitment is crucial to our trust measure. We say there are two different levels of commitments: (1) *norms* (or the implicit commitments), which represent restrictions on the agent's behaviour due to the simple fact that an agent accepts to play a role in a given interaction model; that is, the agent commits to following the rules of the interaction model, and (2) *agreements* (or explicit commitments), which are explicit additional commitments made by the agent at the beginning of the interaction. We represent both types of commitments, norms

and agreements, as:

$$Commit(\beta, \alpha, \langle im, r, m, \varphi \rangle, t)$$

where β commits to α (another agent) to play role r in interaction model im and, as part of playing that role, to send a particular message m instantiated as φ. In other words, φ represents the message that β is committing to. For example, instead of committing to dropping off people at a given location, the firefighter simply commits to finding some other colleague who he can delegate the task to.[7] The commitment is made at time t. Making such a commitment results in α expecting β to later on execute im, playing role r, and instantiating message m as φ (or $\varphi : m$). The trustworthiness of β in α's view will be determined by how β keeps its commitments.

The motivation behind the definition above is that in an interaction, actions could either be message passing actions, illustrating the communication between agents, or internal agent actions, representing all other actions an agent can perform when it is not communicating with others (such as driving a truck, making calculations, etc.). We refer to the latter as the *constraints* that the agent needs to fulfil. We note that the only verifiable (or observable) actions of interactions in distributed open systems are usually the message sending actions.[8] This is because there is no control on how agents will execute internal actions (or constraints) locally. Hence, we say the action that an agent may observe is the sending of the message m. As a result, an agent should commit to the sending of a message m, possibly in a specific format (or instantiated as) φ. In general, however, we say that if agents may commit to and observe any action (including the execution of constraints), then m and φ could resemble any such action, as opposed to message passing actions only.

Nevertheless, while most interaction models might agree that only message sending actions are observable, not all interaction models agree that an agent should commit to sending specific messages. For instance, in LCC, even though the only observable actions are the message passing actions, agents commit to constraints as opposed to messages (the sending of messages is a mere result of executing constraints locally). As an example, the sensor agent in the interaction model of Figure 1 is essentially committing to detect the water level at a given location ($detect(Location, Level)$). By agreeing to engage in this interaction model, the message $water\_level(Location, Level)$ will automatically be sent to the police when the measurement is made. In LCC, agents' commitment to constraints may be viewed as an indirect commitment to a message passing action, since the execution of constraints directly influences the format of the messages being passed. For example, after the constraint $suitable\_loc(Entity, Location)$ is fulfilled, the message $water\_level(Location)$ will automatically be instantiated with the chosen location and sent to the sensor. Hence, in the case of the LCC language, commitments may be modified as follows:

$$Commit(\beta, \alpha, \langle im, r, m, \varphi, c_m \rangle, t)$$

[7]Note that instantiations illustrate how a specific action is executed in reality. As the example above illustrates, instantiations may or may not follow an agent's refinement function $<$. The refinement function, on the other hand, is used to understand the semantic distance between terms, regardless of agents' actions and their chosen instantiations.

[8]By verifiable, or observable actions, we mean actions that agents can assess by simply checking the executed interactions. For example, actions verifiable in the real world, such as delivering goods, cannot be verified against the interaction model's specification. The interaction model may ask the sender agent to inform the receiver agent of the delivery of goods. When performing an automated diagnosis of the sender agent's performance, one cannot confirm whether the goods have actually been physically delivered, but only whether the sender has abided to the interaction model and did not violate its rules, or whether the receiving agent is satisfied with the sender's performance (since agents can rate each other).

where c_m is the set of constraints being committed to, and $\varphi{:}m$ is determined by the set of constraints c_m. We note that the remainder of this paper will use the latter format of commitments.

Last, but not least, we remind the reader that our current implementation makes use of the LCC specified context. As such, norms are considered to be specified by the LCC interaction model itself, while agreements are varying instantiations (or possibly modifications) of different terms of the interaction model. However, we would like to point out that the proposed trust model of this paper may easily be applied to other systems in which the context is specified in a different approach than LCC. Service level agreements (SLA) [Lamanna et al. 2003] are then one alternative method that may be used for formalising commitments. The only requirement imposed by our model is that the context (defined by both norms and agreements) may be translated into a set of commitments, as described in this section.

3.1.3. Observations. For calculating trust, it is crucial to observe the results of previous commitments and decide whether they have been honoured or not. In practice, this implies that for any observable action we want to be able to automatically determine whether the observable action maps to the agent's corresponding commitment. To do this, we assume the existence of a function $g : MSG \times OBS \mapsto \{\top, \bot\}$ that checks whether the expected messages of the set MSG map to the observed messages of the set OBS or not, returning true ($\top$) or false ($\bot$), respectively. Note that for a given interaction im, the set MSG is defined accordingly: $MSG(im) = \{\varphi | Commit(\_,\_\langle im,\_,\_,\varphi,\_\rangle,\_)\}$. The function $g(\varphi,\varphi')$ then checks whether $\varphi \in MSG$ has been satisfied by comparing it to the observed message $\varphi' \in OBS$.

Additionally, we say that the execution of an interaction can possibly generate new commitments for future behaviour in ulterior interactions. For example, agents in one scenario may agree on the product to buy and at what price, and then commit on some conditions for the ulterior negotiation of the payment method: For instance, one agent may say "I'll not charge any bank commissions if you decide to pay by credit card". Hence, one can naturally think of the execution of one interaction as an 'operation' that consumes commitments[9] and generates new ones. This leads us to our basic unit representing the observation made by an agent α about the behaviour of agent β in a given context (described through an interaction model im), which is:

$$\mu = \langle \beta, im, \{\langle Commit(\beta,\alpha,\langle im,r,m,\varphi,c_m\rangle,t),\varphi', g(\varphi,\varphi'), d\rangle\}_{\varphi \in MSG(im)}, C\rangle$$

where β engages in the execution of im with a set of commitments $\{\langle Commit(\ldots),\ldots\rangle\}_{\varphi \in MSG(im)}$, and C is the new set of commitments generated during the execution of interaction im.[10] Note that the third element of the tuple μ describes a set of tuples, each of which records the observed performance of β for a particular message φ committed to in interaction im ($\varphi \in MSG(im)$). We also note that the observable execution of φ is denoted by φ', α's subjective evaluation of β's performance in executing φ is denoted by $d \in D$ (for instance, D may be the qualitative set $\{bad, neutral, good, v.good\}$), and the objective evaluation of β's performance is automatically calculated by the function $g(\varphi,\varphi')$ (where the range of g may be the numerical scale $[0,1]$).[11]

[9]There might be more than one commitment affecting different messages of an interaction.

[10]If one assumes that an interaction im may be divided into a set of sub-interactions $\{im_1,\ldots,im_n\}$, then we have $\mu = \langle \beta, im, \{\langle Commit(\beta,\alpha,\langle im_i,r,m,\varphi,c_m\rangle,t),\varphi', g(\varphi,\varphi'), d\rangle\}_{m \in \{messages(im_i) | im_i \in im\}}, C\rangle$, where the set of initial commitments is the set of commitments corresponding to each message m of each sub-interaction im_i.

[11]The definition above assumes user feedback to be on the level of messages. However, it is more realistic (and practical) to have users give feedback on entire interactions. In such cases, this feedback will then

We denote the set of all existing μs, which we refer to as experiences, as M. For simplification, in the rest of the document we will often view any of the recorded experiences from the perspective of a single message commitment: in other words, we will abuse notation and write $\mu = (\varphi', \varphi)$ to describe the experience when φ was committed to and φ' was observed. We will note the database of experiences of agent α as M_α.

3.1.4. Experiences. This section addresses the issue of how are databases of experiences built. We say there are two types of past experiences, resulting in two different methods of obtaining them: (1) *personal experiences*, which are the agent's personal and trusted experiences that are obtained through observations (introduced above), and they are dealt with by the agent as if they are facts; and (2) *gossip*, which are obtained through the transmission of experiences between agents. An example of a gossip is the following message, in which agent α informs agent γ about its past experience with agent β (in the interaction labelled $negotiation$), where β committed to α to sending a bottle of wine, but instead sent a bottle of whiskey which kept agent α satisfied with the interaction and rating β's performance as $Very\_good$:

$$\begin{aligned} Gossip(\alpha, \gamma, \langle \beta, negotiation, \\ \{\langle Commit(\beta, \alpha, negotiation, seller, \\ send(product), send(wine), \{product(wine)\}\rangle, 3/12/07), \\ send(whisky), \bot, Very\_good\rangle\}, \{\}\rangle) \end{aligned}$$

We note that gossip may be viewed as representing some notions of reputation, i.e. a way of sharing the group's opinion. With gossip, an agent in the network passes information about a previous experience, that is, a particular μ. The main problem in this approach is the reliability of the source. Thus, the question is: given a piece of information $\mu = (\phi', \phi)$ passed from agent β to agent α, what does α think of β's reliability in assessing the experience μ, defined as $\mathbb{R}^t(\alpha, \beta, \mu)$? One way of calculating this reliability measure can be based on social network analysis as exploited in the REGRET system [Sabater and Sierra 2002]. We do not explore this in detail here as it is outside the scope of this paper.

3.1.5. Similarity Measures. Our trust measure is based on using concrete past commitments over $\langle im, r, m, \varphi, c_m\rangle$ (recall that a commitment is defined as $Commit(\beta, \alpha, \langle im, r, m, \varphi, c_m\rangle, t)$) and their results to update the expectation of future behaviour over semantically close commitments. Hence, we need to define similarities between commitments. In our example, we say interaction models (im) may be tagged with a set of keywords or terms from O. Roles (r), messages (m), messages committed to (φ), and constraints committed to (c_m) are usually specified as (possibly complex) terms or simple keywords from O. Hence, in what follows, we first define our general equation for computing similarity between terms and keywords. This is followed by the equations needed for computing the similarity between the various elements of commitments (im, r, m, φ, and c_m). Finally, the equation used to compare two commitments is presented.

Similarity between Terms/Keywords.. The concepts within an agent's ontology are closer, semantically speaking, depending on how far away are they in the structure defined by the '$<$' relation. The measure we use [Li et al. 2003] calculates the *semantic similarity* between two concepts based on the path length induced by '$<$' (more distance in the '$<$' graph means less semantic similarity), and the *depth* of the subsumed concept (common ancestor) in the shortest path between the two concepts (the

be propagated to the different commitments over the different messages, so that the performance w.r.t any message is assumed to be of the same value as the performance of the whole interaction im.

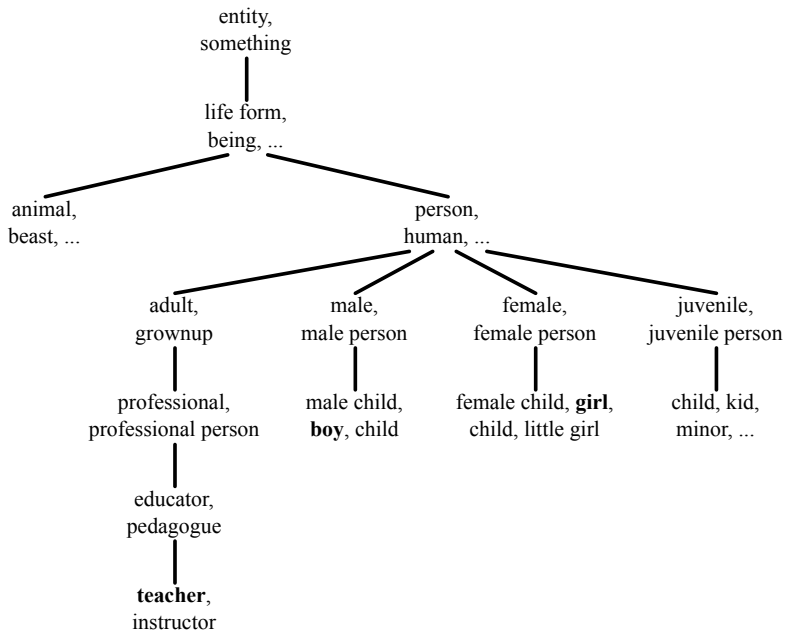

Fig. 2. A portion of an ontology, as presented by [Li et al. 2003]

deeper in the hierarchy, the closer the meaning of the concepts). For example, considering the fragment of the ontology presented by Figure 2, we notice that the shortest path between boy and $girl$ is $boy - male - person - female - girl$ with a length of 4, and that minimum path length between boy and $teacher$ is 6. Thus, we could say $girl$ is more similar to boy than $teacher$ is to boy. When multiple paths may exist between two terms/keywords, then only the shortest path is used in calculating semantic similarity. However, we note that words at upper layers of the ontology have more general concepts and less semantic similarity between words than words at lower layers. Therefore, in addition to the shortest path between terms/keywords, [Li et al. 2003] also considers the depth of the deepest concept subsuming both concepts. In the above example, the deepest concept subsuming both boy and $girl$ is $person, human, ...$, and its depth is 2.

As such, and following [Li et al. 2003]'s proposal, we say that for agent α, the semantic similarity between the terms/keywords θ and θ' is then defined as:

$$Sim1(\theta, \theta') = e^{-\kappa_1 l} \cdot \frac{e^{\kappa_2 h} - e^{-\kappa_2 h}}{e^{\kappa_2 h} + e^{-\kappa_2 h}} \tag{2}$$

where l is the length of the shortest path between the concepts, h is the depth of the deepest concept subsuming both concepts, and κ_1 and κ_2 are parameters scaling the contribution of shortest path length and depth, respectively. Essentially, κ_1 and κ_2 are parameters that α could use to customise the weight given to l and h, respectively. The function $Sim1$ is symmetric (i.e. $Sim1(\theta, \theta') = Sim1(\theta', \theta)$), and its range is $[0, 1]$. In fact, the symmetric nature and the range $[0, 1]$ are properties of all the similarity functions presented in this section (Equations 2–5).

Finally, we note that we provide Equation 2 just as an example, and that our work does not define semantic similarity, but reuses existing approaches. As such, we refer the interested reader to [Li et al. 2003] for further details on Equation 2, and we stress

that alternative approaches can easily be used to replace this equation. There is no universal measure for semantic similarity, and this usually depends on the structure of the ontology amongst other things. Different contexts and different ontologies may require different approaches and equations. Similarly, different agents may also prefer different equations for their own ontologies. This is outside the scope of this paper and deserves a dedicated line of work.

Similarity between Elements of Commitments.. Recall that commitments are usually made with respect to a given interaction, about playing a specific role, sending specific messages, etc. As such, before illustrating how commitments may be compared and their similarity measured, we first illustrate how to measure the similarity between elements of a commitment (i.e. the interaction model im, the role r, the message m, the committed message φ, or the committed constraints c_m). We note that each element of a commitment, say the interaction model im, may be defined through more than one term. For example, to measure the similarity between two interaction models, we say one needs to measure the similarity between all the keywords of the first with all the keywords of the second, and vice versa; and only the terms resulting with maximum similarity are then considered. As such, the similarity between elements of commitments becomes:

$$Sim2(x,x') = \frac{1}{2} \cdot \left(\sum_{\phi \in terms(x)} \frac{\max\limits_{\phi' \in terms(x')} \{Sim1(\phi',\phi)\}}{\mid terms(x) \mid} + \sum_{\phi \in terms(x')} \frac{\max\limits_{\phi' \in terms(x)} \{Sim1(\phi',\phi)\}}{\mid terms(x') \mid} \right) \quad (3)$$

where $(x,x') \in \{(im,im'),(r,r'),(m,m'),(\varphi,\varphi'),(c_m,c'_m)\}$.

For example, consider two interaction models $im1$ and $im2$, where the keywords describing the first are $terms(im1) = \{e\text{-}response, flood\}$ and the keywords describing the second are $terms(im2) = \{emergency\}$. The similarity between the interaction models becomes:

$$\begin{aligned} Sim2(im1,im2) = \tfrac{1}{2} \cdot \big(& \frac{\max\{Sim1(e\text{-}response, emergency)\}}{2} + \\ & \frac{\max\{Sim1(flood, emergency)\}}{2} + \\ & \frac{\max\{Sim1(emergency, e\text{-}response), Sim1(emergency, flood)\}}{1} \big) \end{aligned}$$

In other words, Equation 3 essentially states that the average of the similarity measures of the first set of keywords (or terms) with respect to the second set of keywords (or terms) is considered. As the example above highlights, and to maintain symmetry for the similarity function (i.e. $Sim2(x,x') = Sim2(x',x)$), the average should be repeated to consider the similarity measures of the second set of keywords (or terms) with respect to the first.

But what is the motivation behind choosing this approach for calculating $Sim2$? The basic idea behind this approach is that when considering the similarity of two entities, we need to consider how do the elements composing each entity relate to that entity. For example, is the entity composed of a disjunction of elements, a conjunction of elements, or something in between conjunction and disjunction? In other words, if the keywords describing an interaction model are $\{e\text{-}response, flood\}$, does this mean that the interaction model cannot be described except by *all* of these keywords and not just a subset (i.e. a conjunction of keywords is needed)? Or does this mean that *any* of these keywords is sufficient in describing the entire interaction model (i.e. a disjunction of keywords is needed)? Or is it more like something in between those two cases (i.e. an average of keywords is needed)? In mathematical terms, the average falls in between the conjunction and disjunction, each category can have various degrees, and

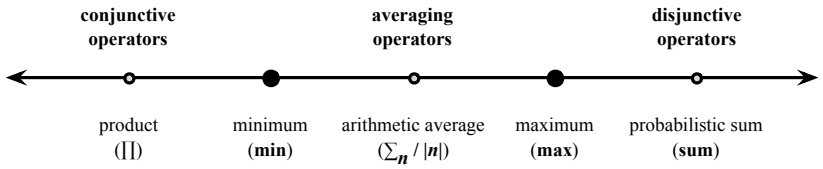

Fig. 3. The classification of operators as conjunctive, average, and disjunctive

different operators are used to express those degrees, as illustrated by Figure 3. Note that the minimum operator ($\min$) separates between conjunctive operators and average ones, while the maximum operator ($\max$) separates between disjunctive operators and average ones. In the Equation 3, when considering keywords describing an entity, we argue that we want to capture the notion of what the *average* of these keywords conveys to us. We do not want to treat them neither as a set of conjunctive terms nor as a set of disjunctive terms, as both would be too restricting. However, in scenarios where a conjunction or a disjunction would better capture the notion of these terms, then the average operator ($\sum_{\phi \in terms(x)} \ / \ | \ terms(x) \ |$) may easily be replaced with another conjunctive or disjunctive operator appropriately, such as the minimum ($\min$) for a conjunction or the maximum ($\max$) for a disjunction.

For instance, in the case of LCC, a constraint c_m may represent a conjunctive set of sub-constraints (e.g. $drop\_off(Entity) \wedge suitable\_loc(Entity, Location)$ of the interaction model of Figure 1). In this case, and as illustrated above, we modify Equation 3 appropriately, replacing the average operator with the minimum.

$$Sim2(x, x') = \frac{1}{2} \cdot \left(\min_{\phi \in x} \{ \max_{\phi' \in x'} \{ Sim1(\phi, \phi') \} \} \ + \ \min_{\phi \in x'} \{ \max_{\phi' \in x} \{ Sim1(\phi, \phi') \} \} \right) \tag{4}$$

Of course, using the minimum operator describes an optimistic approach. For instance, if we are comparing $a \wedge b$ to c and $Sim(a, c) = 0.3$ and $Sim(b, c) = 0.2$, then we have $\min\{Sim1(a, c), Sim1(b, c)\} = 0.2$. For a more pessimistic approach, one can even replace the minimum operator ($\min$) with the product operator ($\prod$). In this case, $\prod\{Sim1(a, c), Sim1(b, c)\} = 0.06$, which is drastically smaller than considering the minimum. Note that in this example, we are only considering the minimum from the point of view of the first constraint ($a \wedge b$). To maintain symmetry (i.e. $Sim2(x, x') = Sim2(x', x)$), and as illustrated by Equation 4 above, one also needs to consider the minimum from the point of view of the second constraint (c).

Similarity between Commitments.. Finally, we define the similarity between two commitments as an aggregation of several similarity measures (note that for simplification, we refer to a commitment $Commit(\beta, \alpha, \langle im, r, m, \varphi, c_m \rangle, t)$ simply as $\langle im, r, m, \varphi, c_m \rangle$):

$$\begin{aligned} Sim \ & (\langle im, r, m, \varphi, c_m \rangle, \langle im', r', m', \varphi', c'_m \rangle) = \\ & Sim2(im, im')^{\gamma_{im}} \cdot Sim2(r, r')^{\gamma_r} \cdot Sim2(m, m')^{\gamma_m} \cdot Sim2(\varphi, \varphi')^{\gamma_\varphi} \cdot Sim2(c_m, c'_m)^{\gamma_{c_m}} \end{aligned} \tag{5}$$

Equation 5 essentially models the aggregation as a weighted combination depending on parameters γ_{im}, γ_r, γ_φ, γ_m and γ_{c_m}. In other words, the parameters are used to help customise the weight given to each part of the commitment. For instance, in some cases, the agent might simply be interested in whether the description of the interaction models and roles are similar (i.e. setting γ_r, γ_φ, γ_m and γ_{c_m} to 0). In other cases, the agent might be interested to give more weight to the exact capabilities of the agent, described through their constraints (i.e. giving more weight to γ_{c_m} than the others). However, how the values assigned to these weights are decided by the agent performing the computation is outside the scope of this paper.

In the remainder of this paper, we will simplify notation and will represent $Sim(\langle im, r, m, \varphi, c_m\rangle, \langle im', r', m', \varphi', c_m\rangle)$ by $Sim(\varphi, \varphi')$.

3.2. Calculating Expectations

As illustrated earlier by Equation 1, calculating the expectation (or probability) of an agent's future performance in performing φ' given that it committed to φ is defined as $P(\varphi_i|\varphi) = P(Can(\varphi_i)) \cdot P(Does(\varphi_i)|Committed(\varphi))$. In what follows we illustrate how $P(Can(\varphi_i))$ and $P(Does(\varphi_i)|Committed(\varphi))$ may be assessed, respectively. This is then followed by a brief discussion of how the general expectation $P(\varphi_i|\varphi)$ loses its value with time.

3.2.1. Calculating Abilities. We say experience will tell us in which different contexts the agent has been *capable* of performing a given action. In the case of LCC, where commitments are made on fulfilling constraints, agent α can define the constraints it knows agent β can deal with as:

$$H_\beta = \{c \mid c \in c_m \wedge \langle \beta, \_, \{...\langle Commit(\beta, \_, \langle im, r, m, \varphi, c_m\rangle, \_), \varphi, 1, \_\rangle ...\}, \_\rangle \in M_\alpha\} \quad (6)$$

The above states that the set of constraints that agent β is known to perform are those constraints that β has committed to in the past and fulfilled (where the performance has been automatically computed by the function g and its value is 1, which represents a successful performance).

Thus, given $Commit(\beta, \alpha, \langle im, r, m, \varphi, c_m\rangle)$, α needs to assess whether β is capable of doing φ. We propose to look into the history of past experiences for all the actions that have been performed. Then, we compute the similarity between current commitments and previously satisfied commitments. In the particular case of LCC, the commitment is on the constraints c_m (recall that c_m could represent a set of constraints, as opposed to a single constraint), hence we say:

$$P(Can(\beta, \langle im, r, m, \varphi, c_m\rangle)) = \min_{\phi \in c_m} \{ \max_{\phi' \in H_\beta} \{Sim(\phi', \phi)\}\} \quad (7)$$

To motivate our choice behind this equation, we remind the reader that what is needed here is to aggregate the similarities between the current constraint and the set of past constraints. The question then arises about which aggregation function to use. Since LCC constraints are composed of a conjunction of terms, we believe a conjunctive operator is needed. We choose the minimum operator ($\min$) for a more optimistic approach. The product operator ($\prod$) can easily replace the $\min$ operator is a more pessimistic approach is required. (Section 3.1.5, especially the subsection entitles "Similarity between Elements of Commitments" along with Figure 3, has provided a thorough discussion on choosing such an operator and the pessimistic/optimistic nature of each.)

3.2.2. Calculating Willingness. We now move to assess $P(Does(\varphi_i)|Committed(\varphi))$, which compares the current context (φ_i, φ) to previous experiences. Suppose that α has an experience $\mu = (\phi_i, \phi)$. Now assume that the μ implies that $P(Does(\varphi_i)|Committed(\varphi))$ should take a new value (say, $R^t(\alpha, \beta, \mu)$, where t represents the current time). However, before learning about the experience μ at time t, α used to think that $P^{t-1}(Does(\varphi_i)|Committed(\varphi)) = \vec{p}$. The question then is: how much credibility (or influence) should the new experience have on calculating the new value $P^t(Does(\varphi_i)|Committed(\varphi))$? To answer this question, we need to compare the previous commitment ϕ to the current commitment φ, the previous commitment ϕ to its observed execution ϕ_i, and the current commitment φ to its expected observed execution φ_i. Then the similarity between the previous context and the current context (or its degree of influence, to be more precise) becomes:

$$S(\phi_i, \phi, \varphi_i, \varphi) = (1 - |Sim(\phi_i, \phi) - Sim(\varphi_i\varphi)|) \cdot Sim(\varphi, \phi) \quad (8)$$

where $Sim(\_,\_)$ follows Equation 5. For instance, take the overly simplified example where a police agent once promised to find a suitable safe location for residents ($suitable\_loc(Residents, Location)$, which we simply refer to as r) and delivers a suitable safe location for students ($suitable\_loc(Students, Location)$, which we refer to as s), and the same agent is now promising to find a suitable safe location for residents ($suitable\_loc(Cattle, Location)$, which we simply refer to as c), and we need to calculate the probability of delivering a suitable safe location for goats ($suitable\_loc(Goats, Location)$, which we refer to as g). If the similarity between what it promised and delivered in the past is equal to the similarity between and it is promising and might deliver now, then the influence of this past experience on this new experience is $Sim(c, r)$ (since $Sim(s, r) = Sim(g, c) \Rightarrow Sim(s, r) - Sim(g, c) = 0$). However, if the similarity between what it promised and delivered in the past is *not* equal to the similarity between what it is promising and might deliver now, then the influence of the past experience on the new experience will lessen the measure $Sim(c, r)$ by taking this difference into consideration ($Sim(c, r)$ is essentially decreased by an amount decided by $Sim(s, r) - Sim(g, c)$).

The new $P(Does(\varphi_i)|Committed(\varphi))$ is then calculated as follows:

$$P^t(Does(\varphi_i)|Committed(\varphi)) = S(\phi_i, \phi, \varphi_i, \varphi) \cdot R^t(\alpha, \beta, \mu) + (1 - S(\phi_i, \phi, \varphi_i, \varphi)) \cdot \vec{p} \quad (9)$$

where $\vec{p}$ describes the past expectation of $P^{t-1}(Does(\varphi_i)|Committed(\varphi))$ that did not take the new experience μ into account, $R^t(\alpha, \beta, \mu)$ describes the expectation based on the single personal experience μ alone, and $S(\phi_i, \phi, \varphi_i, \varphi)$ decides which of the previous two measures should have more influence on the new expectation $P^t(Does(\varphi_i)|Committed(\varphi))$ (i.e. it specifies the weight given to each of those two measures when aggregating them). As for $R^t(\alpha, \beta, \mu)$, it is calculated as follows:[12]

$$\begin{aligned} &R^t(\alpha, \beta, \mu) = \zeta \cdot \vec{p}_\mu + (1 - \zeta) \cdot \vec{p} \\ &\zeta = 1 - \mid Trust^{t-1}(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle) - d \mid \end{aligned} \quad (10)$$

where $Trust^{t-1}$ represents α's trust in β with respect to the current commitment (see Section 3.3), d is α's subjective assessment of the current observation μ (which was defined earlier in Section 3.1.3), and $\vec{p}_\mu$ is the probability distribution describing the value d (an example on how to translate a number in the range $[0, 1]$ into a distribution is presented by [Pinyol et al. 2007]; naturally, other approaches may also be used).

Equation 10 essentially states that R^t takes the value of the probability distribution describing d ($\vec{p}_\mu$) when its previous trust measure ($Trust^{t-1}$) is equivalent to its assessment of the currently observed new experience μ (d). Otherwise, R^t becomes an aggregation of the probability distribution describing d ($\vec{p}_\mu$) and its past expectation that did not consider the new experience μ ($\vec{p}$), where the difference between its previous trust measure ($Trust^{t-1}$) and its assessment of the currently observed new experience (d) decides how much weight is given to each of $\vec{p}_\mu$ and $\vec{p}$.

In summary, R^t reflects α's level of personal *caution* with what this experience means for the future. The motivation behind this equation is that if the results of the current observation and the past trust score are very different, then the experience is perhaps a mistake and should not be taken too seriously, or too strongly into account. However, if there is a slight deviation of the current expected value, this might indicate a tendency in behavioural change.

3.2.3. Decay. $P(\varphi', \varphi)$ is calculated following Equation 1 by multiplying the results of Equations 7 and 9. However, we say the integrity of percepts decreases with time. In

[12] If the experience is not personal, but obtained through gossip, then other mechanisms are used for calculating $R^t(\alpha, \beta, \mu)$, such as those of the REGRET system [Sabater and Sierra 2002].

summary, everything should lose its value, and decay towards some default value (like the uniform distribution). We refer to this default value as the *decay limit distribution*.

Calculating the decay limit distribution is outside the scope of this paper, although we argue that α may have background knowledge concerning the expected integrity of a precept as $t \to \infty$. Such background knowledge will be expressed in terms of α's own knowledge, and is represented as a *decay limit distribution* $\mathbb{D}(X_i)$, where X_i describes the specific context (φ', φ), or the situation in which an agent promises φ and delivers φ'. If the background knowledge is incomplete then one possibility is for α to assume that $\mathbb{D}(X_i)$ has maximum entropy whilst being consistent with the data.

In summary, given a distribution, $P(X_i)$, and a decay limit distribution $\mathbb{D}(X_i)$, $P(X_i)$ decays by:

$$P^{t+1}(X_i) = \Delta_i(\mathbb{D}(X_i), P^t(X_i)) \tag{11}$$

where Δ_i is the *decay function* for the X_i satisfying the property:

$$\lim_{t\to\infty} P^t(X_i) = \mathbb{D}(X_i)$$

For example, Δ_i could be linear:

$$P^{t+1}(X_i) = (1 - \nu_i) \times \mathbb{D}(X_i) + \nu_i \times P^t(X_i)$$

where $\nu_i < 1$ describes the decay rate.

Additionally, one might also think of either the decay function or the decay limit distribution to be also a function of time: Δ_i^t and $\mathbb{D}^t(X_i)$.

3.2.4. Initialisation. It may be argued that in distributed open systems, new agents with no performance history will have low chances to be chosen, compared to those with a good performance history. This is definitely an important issue in open systems. However, we believe that if all agents kept selecting the top agents (or the most trustworthy ones) for collaborating with, then either these selected agents will become more expensive, or they will become a bottleneck that cannot keep up with the demand. In other words, if the top agents are always selected by all others, then with time, these top agents will start providing services of less and less quality, as they may not be able to keep up with the demand. This automatically opens the doors for other agents to be chosen, since we assume agents to consider a variety of aspects when selecting its future collaborator, such as its cost, the quality of the service provided, the efficiency of the service, and so on.

We also note that a new agent's expected performance (P^0) is initially set to the decay limit distribution $\mathbb{D}$, before P^t starts getting shaped by the agent's actual performance. One way to further encourage interacting with new agents, is to set the initial probability distribution describing a new agent's expected performance (P^0) to a distribution that describes above average performances. Of course, this raises other security issues, such as the exposure to whitewashing attacks, where malicious agents re-enter the system with different identifiers so that their negative history is no longer considered and their new identity allows them to be selected for future interactions. The question then is, do we want the system to encourage new users with no history, or would we rather give more importance to an agent's performance history? Naturally, this is a tradeoff that may be addressed differently for different application scenarios. For instance, the severity of the whitewashing attack changes from one system to another based on how expensive or difficult it is for the same agent to obtain a new identifier. Similarly, the minimum P^0 that provides new agents with a chance to be chosen may also vary from one application to another, based on the minimum requirements of the users of that specific application.

3.3. Trust Equations

3.3.1. Trust Measures. After calculating $P(Observing(\alpha, \varphi')|Committed(\beta, \alpha, \varphi))$ at a given time t, which we simply refer to as $P^t(\varphi'|\varphi)$, the question now is: *How do we interpret such expectations?* Or in other words: *How do we calculate a trust measure given an expectation specified as a probability distribution?*

In what follows, we define three different trust equations that can be implemented, or chosen, depending on the particular personality of the agent. In the first, the expected performance, which we refer to as the expected enactment, is compared to an ideal performance (or ideal enactment), which simply specifies what the ideal outcome would be. In the second, the expected performance is compared to what other outcomes (or enactments) are preferred. In the third, the focus is on the certainty of the new expectation.

(1) *Ideal enactments:* Consider a distribution of enactments that represent α's "ideal" in the sense that it is the best that α could reasonably expect to happen: $P_I^t(\varphi'|\varphi)$. For example, even if β has committed to sending wine, α's ideal outcome would be for β to actually send whiskey instead. Trust is then computed by measuring the relative entropy between this ideal distribution, $P_I^t(\varphi'|\varphi)$, and the distribution of expected enactments, $P^t(\varphi'|\varphi)$:

$$Trust(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle) = 1 - \sum_{\varphi'} P_I^t(\varphi'|\varphi) \cdot \log \frac{P_I^t(\varphi'|\varphi)}{P^t(\varphi'|\varphi)} \tag{12}$$

where "1" is an arbitrarily chosen constant being the maximum value that this measure may have, and φ' represents the potential outcome (or observed results).
It may be questioned whether it is fair to allow agents to set ideal enactments that are different from what was agreed upon, resulting in one having its trust measure be less than perfect even when the it always does exactly what it agreed to do. This may easily be solved by restricting agents from defining their own ideal enactments and assuming an ideal enactment to describe the case when the agent in question simply fulfils its promises. Nevertheless, we say it is also possible (if needed) to permit agents to describe their own ideal enactments because it may also be argued that one can always do better than what they promise, and there should be means for capturing that. For instance, a seller who promises to deliver an item in 7 days and ends up delivering it in 2 is always preferred to a seller who promises to deliver an item in 7 days and does deliver it in 7.
Last, but not least, we note that although we use the equation of relative entropy (Equation 12) to measures the distance between two probability distributions, we note that alternative methods for calculating this distance may also be considered. For instance, Equation 12 may be replaced by the earth mover's distance ($Trust(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle) = 1 - EMD(P_I^t(\varphi'|\varphi), P^t(\varphi'|\varphi))$), where the earth mover's distance (specified through the function EMD) is a measure of the distance between two probability distributions [Rubner et al. 1998].[13]

(2) *Preferred enactments:* This measures the extent to which the enactment φ' is preferable to the commitment φ. It requires α to specify its preferences with respect to enactments via the predicate $\mathrm{Prefer}(c_1, c_2)$, meaning that α prefers c_1 to c_2.

[13] If probability distributions are viewed as piles of dirt, then the earth movers distance measures the minimum cost for transforming one pile into the other. This cost is equivalent to the amount of dirt times the distance by which it is moved, or the distance between elements of the region E. The range of EMD is $[0, 1]$, where 0 represents the minimum distance and 1 represents the maximum possible distance. However, we note that to use the earth mover's distance, one also needs to determine what the distance between the terms of the ordered region E is. That is one needs to define the matrix $D = \{d_{ij}\}_{i,j \in E}$, where d_{ij} represents the distance between the elements i and j of the region E.

$\text{Prefer}(c_1, c_2)$ is then transformed into a normalised probabilistic measure, defined as $P^t(\text{Prefer}(c_1, c_2))$. As such, the final trust measure is then simply an aggregation of the various expected outcomes ($P^t(\varphi' \mid \varphi)$), where the weight of each is decided by the agent's predefined preferences. This is expressed accordingly:

$$Trust(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle) = \sum_{\varphi'} P^t(\text{Prefer}(\varphi', \varphi)) \cdot P^t(\varphi' \mid \varphi) \tag{13}$$

(3) *Certainty in enactment:* This measures the consistency in expected acceptable enactment of commitments, or in other words, "the lack of expected uncertainty in those possible enactments that are better than the commitment as specified". We use entropy to measure uncertainty, and we note that the minimal the uncertainty (or the minimal the entropy) then the maximal trust is. As such, we say let $\Phi_+(\varphi, \kappa) = \{\varphi' \mid P^t(\text{Prefer}(\varphi', \varphi)) > \kappa\}$ for some constant κ, and:

$$Trust(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle) = 1 + \frac{1}{B^*} \cdot \sum_{\varphi' \in \Phi_+(\varphi, \kappa)} P^t_+(\varphi' | \varphi) \cdot \log P^t_+(\varphi' | \varphi) \tag{14}$$

where $P^t_+(\varphi' | \varphi)$ is the normalisation of $P^t(\varphi' | \varphi)$ for $\varphi' \in \Phi_+(\varphi, \kappa)$, and:

$$B^* = \begin{cases} 1 & \text{if } |\Phi_+(\varphi, \kappa)| = 1 \\ \log |\Phi_+(\varphi, \kappa)| & \text{otherwise} \end{cases}$$

3.3.2. Aggregating Trust Measures. In the previous section, we have illustrated how trust measures may be computed for specific commitments by calculating $P(Observing(\alpha, \beta, \varphi) | Committed(\beta, \alpha, \varphi))$ for a particular im, r, m and φ. If we are interested in trust measures for more general cases, then all past experiences that fit this general case are used. For instance $Trust(\alpha, \beta, \langle im, r \rangle)$ will be computed by looking into all experiences for im and r in the database. Measures that take into account the importance of certain commitments (or ims or rs) are also easy to define. Imagine a normalised function that gives the importance of terms $f : Terms \to [0, 1]$, where $Terms$ is the set of terms built on top of the ontology O. We can then define an aggregation as:

$$Trust(\alpha, \beta, \langle im, r \rangle) = 1 - \sum_{\varphi} P^t_\beta(\varphi) \cdot f(\varphi) \cdot [1 - Trust(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle)] \tag{15}$$

where $P^t_\beta(\varphi)$ is a probability distribution over the space of commitments that the next commitment β will make to α is φ. We note that building $P^t_\beta(\varphi)$ is not a straightforward task; it is something learned over time. We say $P^t_\beta(\varphi)$ may be built by learning what the other agent does. In other words, this equation may be applied when agent α has a good knowledge of the patterns followed by agent β.

3.4. Trust Algorithm

We now give, as an example of a default trust algorithm, one that uses the 'preferred enactments' trust equations (Algorithm 1). We assume decay is linear and the decay limit distributions are equiprobable distributions. Other algorithms may be similarly defined. A generic version of the algorithm, where functions like $Sim(\cdot)$ or distributions like $\mathbb{D}(\cdot)$ are parameters, is also straightforward.

The algorithm has parameter η that determines how much of the semantic space is explored. By fixing it to a high value, we can have more efficient implementations. By reducing it progressively, we can have a more realistic and fine grained implementation. Also, techniques like memorising can help in increasing the efficiency of the algorithm.

Trust is calculated *on demand* following Algorithm 1. Other implementations that pre-compute probability distributions are possible but not considered here.

We note that the complexity of the trust algorithm is linear over the following constants: (1) the size n of an agent's ontology; (2) the size m of an agent's history of past experiences; and (3) the size p of the dialogue (or interaction model) in question. The algorithm has several loops that go over the elements of the ontology, the history of past experiences, and the dialogue. However, the most complex loop is the final one, which calculates the trust measure. This loop iterates over elements of the agent's history m and elements of the dialogue p (in terms of commitments made by a given dialogue). The loop then contains two sub-loops that iterate over the agent's ontology n. The complexity then becomes $\mathcal{O}(m \cdot p \cdot n)$. As such, we that for reasonably sized ontologies and dialogues, trust can then be computed in real time as the history can always be limited to the most recent experiences.

4. MATCHING ALGORITHM

The capability of an agent has been assessed in the previous section through observing whether or not an agent did in fact perform the action in question in some past experience. This information, however, is not always enough. Agents' capabilities are dynamic and continuously evolving. Agents may want to take on roles similar - or even rather different - roles to those they have already played. Different IMs (see Section 2 for an explanation of these) may be used, so that a task that is essentially the same as they have previously played may have slightly different data matching requirements. Therefore, even if the trust algorithm can give useful information about an agent's ability to perform a particular task in general, we require more information to help us decide if the agent can meet the technical specifications of the current implementation. This is provided by the matching algorithm.

We present a method for obtaining a value $M \in [0, 1]$, which describes the ability of an agent to perform a role (where the value 0 would describe complete disability, and the value 1 would describe full ability). This value is calculated by the agent itself to determine whether or not it wishes to perform the role (and, if so, how best to do that). The value can then be shared with other agents which may be considering interacting with it, so that they can see how the agent evaluates its own ability to perform that role.

An obvious drawback of this process is that it is not possible to verify the score that an agent gives of its own ability. The mappings are between the requirements of the role, which are public, and the particular abilities of an agent, which are private; hence, they cannot be replicated by any other agents. However, this is valuable information about an agent's ability to perform a role which is highly pertinent when choosing agents to interact with and which cannot be drawn from any other source, so even though it may be unreliable it is still potentially very useful information. Fortunately, this problem can be mitigated against to a large extent by combining these matching scores with trust scores. An agent that frequently oversells its ability to perform a role will frequently under-perform, and thus have low trust scores associated with it. An agent that has a high trust score can be assumed to generally give honest information about its ability.

Within an IM, the ability to perform *role* is determined purely through the ability to satisfy constraints on that role (as explained in Section 2). The agent wishing to perform that role must therefore ascertain whether it can solve those particular constraints. If an agent has been specifically designed for a given specification of a role, it can be assumed that it is capable of adequately performing those tasks. But if an agent is attempting to perform a different specification of the role it usually plays, or a

ALGORITHM 1: Calculating $Trust(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle)$

Input: O (agent's finite local ontology), $\kappa_1, \kappa_2 : Real$ [Default 1.0] (parameters of the similarity function, Equation 2), $\eta : [0, 1]$ [Default 0.8] (min. sem. similarity in the computation), $\nu : [0, 1]$ [Default 0.95] (decay parameter), $Prefer : Terms(O) \times Terms(O) \rightarrow [0, 1]$ [Default *Prefer(x, y) = if x = y then 1 else 0*] (a prob. distribution for preference over terms represented as a square matrix), $M_\alpha \subseteq M$ (α's log of experiences sorted by time), $\mathbb{R}^t(\alpha, \beta) : M \rightarrow [0, 1]$ [Default $\mathbb{R}^t(\alpha, \beta, \mu) = 1 - \mid Trust^{t-1}(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle) - d \mid$)] (reliability measure of experiences, by default $D = [0, 1]$).

Output: Calculating the trust measure $Trust(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle)$

$Focus \leftarrow \emptyset$; /* $Focus$ is the set of ontology terms that this run should focus on */

forall the $\varphi' \in Terms(O)$ **do** /* This builds $Focus$ set, assuming $Terms(O)$ is finite */

  if $Sim1(\varphi', \varphi) \geq \eta$ **then**

    $Focus \leftarrow Focus \cup \{\varphi'\}$;

  end

end

forall the $\varphi' \in Focus$ **do** /* This defines the decay limit distribution D */

  $D(\varphi' \mid \varphi) \leftarrow 1/size(Focus)$;

end

$H_\beta = \emptyset$;

forall the $\mu = \langle \beta, im, PC, C \rangle \in M_\alpha \ \wedge \ \langle Commit(\beta, \_, \langle im, r, m, \varphi', c'_m \rangle, t), \varphi'', 1, d \rangle \in PC$ **do**

  $H_\beta = c'_m \cup H_\beta$; /* This populates the set of all actions H_β that β can perform */

end

$P_{can} \leftarrow 1$;

forall the $\phi \in c_m$ **do** /* This calculates $P(Can(\beta, \varphi))$, specified here as P_{can} */

  $MAX \leftarrow 0$;

  forall the $\phi' \in H_\beta$ **do**

    $MAX \leftarrow \max\{MAX, Sim(\phi, \phi')\}$;

  end

  $P_{can} \leftarrow \min\{MAX, P_{can}\}$;

end

$t \leftarrow 0$; $Trust^t \leftarrow 0$; /* Initially, time is set to 0, and trust score is set to 0 */

$P^t = D$; /* Initially, $P(Does(\varphi) | Committed(\varphi))$, specified as P, is set to D */

forall the $\varphi' \in Focus$ **do** /* The trust score is based on the initial value of P */

  $Trust^t \leftarrow Trust^t + Prefer(\varphi', \varphi) * P^t(\varphi' \mid \varphi)$;

end

forall the $\mu = \langle \beta, im, PC, C \rangle \in M_\alpha \ \wedge \ \langle Commit(\beta, \_, \langle im, r, m, \varphi_c, c_m \rangle, t), \varphi'', \_, d \rangle \in PC$ **do**

  if $Sim(\varphi_c, \varphi) \geq \eta$ **then** /* For each subsequent timestep, and for each past experience that is considered relevant to the current experience in question, the value of P is updated, following Equations 8, 9, and 11 */

    forall the $\varphi' \in Focus$ **do**

      $T \leftarrow (1 - \mid Sim(\varphi'', \varphi_c) - Sim(\varphi', \varphi) \mid) \cdot Sim(\varphi_c, \varphi)$;

      $Q(\varphi' \mid \varphi) \leftarrow P_{can} \cdot (T \cdot (\mathbb{R}^t(\alpha, \beta, \mu) \cdot d + (1 - \mathbb{R}^t(\alpha, \beta, \mu) \cdot P^t(\varphi' \mid \varphi))) + (1 - T) \cdot P^t(\varphi' \mid \varphi))$;

      $P^{t+1}(\varphi' \mid \varphi) \leftarrow (1 - \nu) D(\varphi' \mid \varphi) + \nu \cdot MRE(P^{t-1}, Q)$; /* MRE is the Minimum relative entropy distribution from P^{t-1} satisfying Q */

    end

  end

  $t \leftarrow t + 1$;

  $Trust^t \leftarrow 0$;

  forall the $\varphi' \in Focus$ **do** /* The trust score is updated at each timestep */

    $Trust^t \leftarrow Trust^t + Prefer(\varphi', \varphi) * P^t(\varphi' \mid \varphi)$;

  end

end

return $Trust$;

slightly different role, it will need to assess whether or not the tasks in that role are a good match to its abilities by matching the constraints in that role to its own abilities.

This matching will not be a simple word-to-word matching; rather, since constraints are structured, first-order terms, we will require structured matching.

We describe an algorithm, which has been implemented within the OpenKnowledge system, which matches first-order constraints to a first-order description of an agent's abilities. For example, if a WSDL service description is translated into a first-order term (such a translation process is already implemented within the OpenKnowledge system), this process could match expected input with actual input. For each constraint i in the role, the algorithm finds the ability of the agent which best matches it and returns a value $M_i \in [0, 1]$ describing the quality of this match: 1 represents a semantically perfect match, while 0 is returned if there is no matching ability at all. As part of this process, the algorithm also develops a map that maps a constraint to an ability, to help the agent determine how to satisfy this constraint using its ability. The process is repeated for all constraints for a given role in a given interaction. These scores are then combined, and a single value $M \in [0, 1]$ is returned, describing the overall ability of the agent to perform that role. The remainder of this section is dedicated to the calculation of this measure M.

4.1. Introducing Structure-preserving Semantic Matching

Value $M \in$ [0 1] is calculated by the *Structure-preserving Semantic Matching* (SPSM) algorithm [Giunchiglia et al. 2008]. This algorithm is designed to map two trees to one another, so the first step is to convert our first-order terms to trees, with the predicate name becoming the root of the tree and the arguments becoming the children. For example, consider a constraint concerning the reporting on the water-level. One agent role that could use such a constraint could be the role describing the role of a sensor agent in an emergency response flooding scenario. For example, the first-order constraint could be:

`measurement(location(ReporterID,Node),Level,date(Month,Day,Hour,Minute))`

and the first-order ability of the sensor agent might be:

`reading(ReporterID,Node,date(Day,Month,Year,Time),Water-level).`[14]

These would be converted into trees and mapped as illustrated in Figure 4.1.

SPSM allows us to detect *good enough* matches by producing this score $M \in [0, 1]$ which can then be compared with a threshold value. Any match which exceeds this value is considered to be acceptable; any match which is lower than this value is rejected, and the two terms are considered not to match. Since the concept of good enough is very context dependent – in safety critical situation perhaps only a near-perfect match will do but in other situations a much weaker match may suffice – this threshold is set by the user according to the particular interaction [Giunchiglia et al. 2008]. This algorithm can be performed quickly on-the-fly, during run-time.

Since the details of the matching process have already been published elsewhere, we merely outline the process in this section and direct the interested reader to the paper by Giunchiglia et al. [2008] for further information.

4.2. The SPSM algorithm

Matching is performed in two steps: node matching and tree matching. During node matching, all nodes in the first tree are matched to all nodes in the second tree, regardless of their positions within those trees. This is done using adapted conventional

[14]Note that in such formulae, and in Figure 4.1, the names of the variables indicate the types expected, with an initial upper-case letter indicating that they are variables. For example, the second level argument `Level` indicates that the argument is a variable that should be instantiated with a value of type '`level`'.

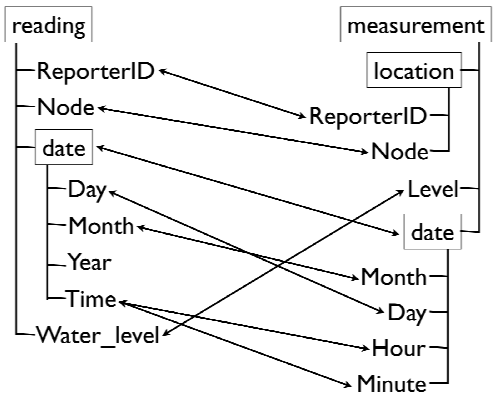

Fig. 4. Two approximately matched first-order terms — T_1:`reading(reporterID,node,date(time,day, month,year),water-level)` and T_2:`measurement(level,location(reporterID,node),date(month,day, hour,minute))` — are specified as trees. Predicates are in rectangles with rounded corners; they are connected to their arguments by lines. Node correspondences are indicated by arrows.

ontology matching techniques, and is based on the S-Match system [Giunchiglia and Shvaiko 2003]. The tree-matching step then exploits the results of the node-matching step to provide a global match for the overall trees, taking into account the structures of the trees.

4.2.1. Abstraction Operations. Our structural matching techniques are based on the theory of abstraction, as developed by Giunchiglia and Walsh [1992]. This describes the three ways in which a first-order term may be more general than another first-order term:

— *Predicate abstraction*: The predicate of term 1 may be more general than the predicate of term 2: e.g. $reading(X)$ is mapped to $water\text{-}reading(X)$
— *Domain abstraction*: One of the arguments within term 1 may be more general than the corresponding argument in term 2: e.g. $reading(Level, Date)$ is mapped to $reading(Water\text{-}Level, Date)$
— *Propositional abstraction*: Term 1 may have fewer arguments than term 2: e.g. $reading(Level)$ is mapped to $reading(Level, Date)$

We invert these to form the equivalent *refinements* (where a refinement is the inverse of an abstraction), and allow, for both matches between predicates and matches between arguments, the possibility of equivalence and of disjunction (no match). This results in the full set of matches that we allow. This set preserves the desirable property that functions are only mapped to functions and variables are only mapped to variables.

4.2.2. Tree Edit Distance via Abstraction Operations. The matching of the trees is done using a tree edit distance algorithm [Tai 1979]. However, we restrict the formulation of the tree edit distance problem in order to reflect the semantics of the first-order terms. We do this by redefining the tree edit distance operations so that they have one-to-one correspondence with the abstraction/refinement operations. Any forbidden alignment can be assigned an infinite cost. This allows us to alter the conditions depending on requirements: for example, if we wish to only allow a perfect match or a more precise

match, then any alignments that include a generalisation can be assigned an infinite cost.

One undesired aspect of tree edit distance matching is that horizontal node ordering is preserved. We do not believe that there is much semantic value to the ordering of arguments in a predicate, and so wish to allow horizontal ordering to be changed as necessary. We facilitate this by allowing reordering of the nodes as desired, without cost.

4.2.3. Global similarity between trees. We calculate an overall cost of mapping one tree (or first-order term) into another using the following equation:

$$Cost = min \sum_{t \in S} k_i * Cost_i \tag{16}$$

where S stands for the set of the allowed tree edit operations; k_i stands for the number of i^{th} operations necessary to convert one tree into the other; and $Cost_i$ defines the cost of the i^{th} operation. Our goal here is to define the $Cost_i$ in a way that models the semantic distance.

The similarity of these two trees (or first-order terms) is then calculated using:

$$TreeSim = 1 - \frac{Cost}{max(T_1, T_2)} \tag{17}$$

where $Cost$ is taken from Equation 16 and is normalised by the size of the biggest tree. Note that for the special case of $Cost$ equal to ∞, $TreeSim$ is estimated as 0.

4.2.4. User- and domain- specific matching. The SPSM algorithm is generic, and does not consider any user- and domain-specific requirements. However, there are circumstances in which such requirements could be usefully incorporated into the matching algorithm.

These user- and domain-specific requirements fall into two types:

— *Preferences about nodes:* for example, when matching *reading(Date,Level)*, it may be very important that *Date* is matched accurately, but *Level* is much less important. These kinds of preferences could be added to the algorithm, by allowing the user to input weightings for each node. If only a perfect match is acceptable for a particular node, an infinite weight can be assigned to it. The calculation of similarity will then multiply the value of similarity between each node by its weighting, and thereby produce an overall matching score which takes such preferences into account. Thus the matching cost for a node with an infinite weighting will only finite if the matching score is zero (i.e., a perfect match).

— *Domain-specific information:* for example, *Water* is a subtype of *Liquid*, so the matching penalty for equating these types would be fairly low (as a subtype relation implies quite a high level of similarity), but non-zero. But perhaps in this situation, the words *Water* and *Liquid* are used more or less interchangeably, so it would be better to have a zero penalty for this match, even though these words would not normally be considered interchangeable. In some domains, this kind of domain-specific equivalence (or possibly domain-specific near-equivalence) may be desirable with words that would not normally be considered to be related at all. A way to incorporate this kind of information would be to allow users to input domain-specific information, such as that *Water* and *Liquid* should be considered equivalent. The matcher would then check the specific information that has been added to see if that will tell it, in this particular domain, how good this match is before it would try its generic matching approaches (such as dictionary look-up). Developing an automated approach that would be able to determine such domain-specific matches

would be very difficult, though perhaps some kind of statistical observation of co-occurrence could help here.

Implementing these extensions is a straightforward task. This paper, however, discusses a generic approach to the problem, and we believe that in most situations the generic approach is adequate and requires less overhead from a human user. Implementing the extensions is left for future work.

5. GOOD ENOUGH ANSWERS (GEA) ALGORITHM

Section 3 has presented a trust algorithm that calculates the trust in an agent to carry out a given action by analysing the agent's past experiences and deducing the probability of the agent to be both capable and willing to perform the action. Section 4, on the other hand, provides a mechanism that allows an agent to calculate its true capability measure for a particular specification by matching the action in question with its own concrete capabilities, and share this measure with others upon their request.

As a result, a final measure on the expected performance of an agent should be based on both the matching and trust scores obtained above. But how may we combine such scores? The trust scores gives us an indication of intention and general ability; the matching score gives us information about whether the precise requirements of the role are a *good enough match* to the agents abilities. We need to combine these to determine whether the agent is likely to provide *good enough answers* in this role: that is, will the information provided by the agent during this interaction - or more generally, the actions the agent is required to take - be of a standard that is likely to be acceptable, and if there are many agents competing to perform the role, which of these would we expect to perform it best?

The *good enough answers (GEA)* algorithm is therefore a tool for an agent (agent 1) intending to interact with other agents within a specific interaction, to allow it to determine which, if any, of those other agents (for example, agent 2) it wishes to interact with. Agent 1 will calculate its trust score for agent 2, and will be able to access the matching score presented by agent 2. It can then use the GEA algorithm to combine these two scores to enable it to estimate how well agent 2 will perform, and then to compare it to the other agents, before commencing (or refusing to commence, if the scores of all other agents are too low) on the interaction. Note that the method of combining these two scores is decided pragmatically rather than theoretically, as there is no fixed way in which this should happen. In what follows, we propose two different methods/algorithms for achieving this:

(1) **Combining matching and trust scores:** This approach, illustrated in Algorithm 2, first removes all agents with a matching score below a certain threshold and then, for the remaining agents, calculates their trust scores. The $best\_agent$ is the one for which the combination of the trust and matching scores is highest, calculated according to the equation $t_\alpha \cdot \nu + m_\alpha \cdot (1 - \nu)$, where t_α is the trust score for agent α, m_α is α's matching score and ν is a parameter. In the simplest case, $\nu = \frac{1}{2}$, in which case this equation represents taking the average of t_α and m_α. The value of ν must be determined empirically. Nevertheless, it may be that a linear combination of the two scores is not the best approach.

(2) **Determining intervals:** This approach, illustrated in Algorithm 3, sorts agents into bands of width ψ ($\psi \in [0...1]$) according to their matching scores. Thus the first band would contain agents with perfect matching; the second band would contain agents with matching scores $(1 - \psi) \leq m_\alpha < 1$, and so on. Once this sorting has been done, matching scores are ignored and further choice is made on the basis of the trust scores alone. The $best\_agent$ is the one from the highest matching band that has the best trust score, assuming that this trust score exceeds some basic

ALGORITHM 2: Agent Selection — Finding the best agent for a role r in interaction model IM by combining matching and trust scores, as calculated by agent β

Input: A (the set of agents subscribed to the role), $\forall \alpha \in A \cdot m_\alpha \in [0,1]$ (where m_α is the matching score declared by α on subscribing to the role r in interaction model IM), $\xi : [0,1]$ (the matching threshold), and $\nu : [0,1]$ (trust/matching weighting variable).
Output: The most suitable agent $best\_agent$ is selected.

```
poss_agents ← ∅;
forall the α ∈ A do
    if m_α ≥ ξ then
        poss_agents ← poss_agents ∪ α;
    end
end
highest_score ← 0;
best_agent ← ∅;
forall the α ∈ poss_agents do
    t_α = trust(β, α, IM, r);
    α_score = t_α · ν + m_α · (1 − ν);
    if α_score > highest_score then
        highest_score ← α_score;
        best_agent ← α;
    end
end
return best_agent;
```

threshold ζ. If the highest trust score for an agent from the highest matching band does not exceed this threshold, these agents are rejected and the next matching band is inspected, until a suitable agent is found.

In developing these algorithms, a pragmatic priority is to minimise the number of agents for which it is necessary to calculate trust scores, because this is time consuming, whereas considering matching scores is trivial, because these are calculated by each agent themselves. If agents can be eliminated on the basis of their matching scores alone, this saves the necessity to find a trust score for them. Trust scores can then be calculated only for those agents that have a high potential of being suitable.

Of course, these two algorithms are only a sample of how one could use both trust and matching scores for selecting a suitable agent. Further testing is needed to evaluate their relative merits on their performance. This could indicate that neither of the approaches is optimal and point us towards an improved method of integration of matching and trust scores.

6. NOTES ON THE REQUIRED PARAMETERS

Before we conclude the presentation of our proposed trust, matching, and GEA models, we discuss their required parameters. We understand that at first sight, the parameters might seem numerous in this proposal, raising questions such as whether the parameters influence each other. However, a careful inspection illustrates that the parameters may be divided into five main independent categories:

— **Similar terms parameters (ST).** These parameters (κ_1, κ_2, and η) control which terms are considered similar in a given ontology. The parameters κ_1 and κ_2 are parameters that one could use when calculating the similarity between terms to customise the weight given to the length of the shortest path between two terms in an ontology and the depth of the deepest term in the ontology subsuming both terms in question (see Equation 2 for details). The parameter η specifies the threshold for

ALGORITHM 3: Agent Selection — Finding the best agent for a role r in interaction model IM through threshold lowering, as calculated by agent β

Input: A (the set of agents subscribed to a role), $\forall \alpha \in A \cdot m_\alpha \in [0,1]$ (where m_α is the matching score declared by α on subscribing to the role r in interaction model IM), $\psi : [0,1]$ (threshold lowering interval), and $\zeta : [0,1]$ (trust threshold).
Output: The most suitable agent $best\_agent$ is selected.

```
poss_agents ← ∅;
threshold ← 1;
highest_trust ← 0;
best_agent ← ∅;
while best_agent = ∅ do
    while poss_agents = ∅ do
        forall the α ∈ A do
            if m_α ≥ threshold then
                poss_agents ← poss_agents ∪ α;
            end
        end
        threshold ← threshold − ψ;
    end
    forall the α ∈ poss_agents do
        t_α = trust(β, α, IM, r);
        if t_α ≥ ζ then
            if t_α > highest_trust then
                highest_trust ← t_α;
                best_agent ← α;
            end
        end
    end
end
return best_agent;
```

similarity measures and it determines how much of the semantic space is explored (see Section 3.4 for details).

— **Interaction's components preference parameters (IC).** These parameters (γ_{im}, γ_r, γ_φ, γ_m and γ_{c_m}) determine the weight given to each element of an interaction model. For instance, they decide whether the user is more interested in similar interactions, roles, or specific actions (see Equation 5 for details).

— **Decay limit distribution (DD).** This parameter ($\mathbb{D}$) describes the default knowledge we form about others' expected performance when there isn't sufficient past experience to build our measure on. The basic idea is that information loses its value with time, and with the lack of new experience, past expected performances lose their value by decaying towards the decay limit distribution $\mathbb{D}$ (see Equation 11 for details).

— **Decay rate parameter (DR).** This parameter (ν) determines the rate of decay. It specifies how fast does the expected performance decay towards the decay limit distribution $\mathbb{D}$ (see Equation 11 for details).

— **Trust/Matching preference parameters (TM).** These parameters (ν and ξ of Algorithm 2, or ψ and ζ of Algorithm 3) describe how the trust and matching scores may be combined. In the first approach, agents whose matching score is below a certain threshold ξ are discarded, and the rest are ranked according to the aggregation of their trust and matching scores, where ν decides the weight given to each. In the second approach, agents are sorted into bands of width ψ, and the agent with

the highest trust score above the threshold ζ from the highest matching band is selected.

Concerning the interdependence amongst the different parameter sets, we note that the parameters of one category do not influence (and are not influenced by) the parameters of another. The only thing the parameters ST affect is defining the semantic space to be explored. A richer semantic space results in more informative computations, and the scarcer the semantic space results in more efficient computations. In any case, the parameters of ST do not impact any of the other parameters.

The parameters of IC influence the aspects that are given a priority when predicting future performance. For instance, one might be interested in comparing performance to similar interactions (say, all e-commerce interactions), without caring about the details of the specific agent actions (say, did the seller deliver the item on time, or was he helpful in answering buyers' questions, etc.). Again, these parameters cannot influence those of ST, DD, DR and TM. However, if they give any aspect a very low weight, then the computation effort spent on that aspect may be wasted.

The decay limit distribution DD defines 'the default' in specific applications (for instance, it states that all agents are by default good, bad, average, or somewhere else in between). As such, this parameter also cannot influence other parameters.

The decay rate DR specifies how quickly information loses its value. This parameter too cannot influence other parameters, as it simply states how quickly is the decay limit distribution DD reached. However, if the decay rate is set to an unreasonable measure where the value of information is lost very fast, then the computational effort spent on computing the trust measure may be wasted. This includes the computational effort plus the effort spent on tuning the parameters of ST and IC.

Finally, the parameters of TM decide how the trust and matching measures are combined, and they cannot influence the parameters used in computing those measures. However, we note that if the trust measure becomes almost negligible (i.e. Algorithm 2 is used and its parameter ν is set to a low value that is close to zero), then the effort spent on computing the trust measure may be wasted. This includes the computational effort, plus the effort spent on tuning the parameters of ST, IC, DD, and DR.

As for how to decide what values are assigned to each parameter, the procedure needed to do so will differ from one domain (or application) to another. Furthermore, outlining such a procedure requires a detailed study of the domain description, and neither does this paper have the space to do so, nor is an exhaustive analysis of such a procedure part of the scope of this research. Nevertheless, in what follows we provide a brief and general set of guidelines that would help set the interested reader in the correct direction for assigning the parameters' values.

First, we note that there is a need to run a number of simulations, each focusing on tuning the parameters of a different category of the above. Then, for each category, the tuning of the parameters will be based on the domain description. For example,

— in domains where ontologies are more specialised, the depth between two terms that is used in calculating the similarity of terms becomes less important than in broader ontologies; in other words, the more specialised an ontology, then the higher the value of parameter κ_2 (we refer the reader to Equation 2 and [Li et al. 2003] for further details on this parameter)
— an agent that scarcely interacts with others will require lower values for the parameter η, which implies that a larger semantic space is explored (we refer the reader to Section 3.4 for further details on this parameter)
— a trusting agent will require a high default expectation (specified through the decay limit distribution $\mathbb{D}$), whereas a distrustful agent will require a low default expectation; in other words, the probability distribution $\mathbb{D}$ must reflect good behaviour

when the agent expects others to behave properly unless proven otherwise, and it must reflect bad behaviour when the agent expects others to behave poorly unless they prove themselves (we refer the reader to Section 3.2.3 for further details on this parameter)
— agents may consider a high decay rate ν in scenarios where agent behaviour is continuously changing, whereas they may consider lower decay rate in scenarios where agent behaviour rarely changes (we refer the reader to Section 3.2.3 for further details on this parameter)

As illustrated above, based on the application and its requirements, the simulations will show which values are appropriate for the given application scenario in question.

It may be noted that cases may arise where simulations are not enough for learning how parameters should be set for a given scenario, since real systems may bring about new challenges. This is natural, and we concur that we cannot prevent such situations from arising. This is in fact an inherent problem in simulation, which can never be guaranteed to cover exactly an unknown and dynamic real-world event, and is not specific to our approach. However, what can be done is to have engineers tweaking the parameters at deployment time accordingly.

7. EVALUATION: CHOOSING AGENTS IN AN E-RESPONSE SCENARIO

We have designed an e-response experiment where various agents try to collaborate to overcome a flood disaster scenario. Amongst the agents are *sensor* agents, which are responsible for measuring the water-level in meters, and the *Civilian Protection Unit*, which collects water level readings from sensor agents at various locations. In this scenario, several agents can play the role of *sensor*; therefore, it is crucial to select the best agent to interact with in order to retrieve the most accurate water level reading at each location. Note that we only provide a summary of our results in this section, while we refer the interested reader to our technical document [Pane et al. 2008], for a more detailed description.

7.1. Selection Strategies

The *Civilian Protection Unit* needs to periodically select the most accurate *sensor* agents every time (denoted as a *timestep*) it wishes to check the water level. There can be different selection strategies which can be adopted by the Civilian Protection Unit. In what follows, we present the three strategies that we choose to test and compare in our experiment.

— Random Strategy: this strategy is the simplest one. The selecting agent (the Civil Protection Unit in our case) first groups the *sensor* agents according to their locations. Then, for each location, it picks up an agent in a random fashion. This strategy does not take into account any information about the behavior assumed by an agent in the past interactions: all agents are regarded as good agents for the interaction to come.
— Trust Strategy: this strategy takes into account the trust scores of the *sensor* agents. As before, all the agents are grouped by location; then, a trust score is computed using information from past interactions and, finally, the agent with the highest trust score is selected for each location. Agents having identical highest trust score are subject to a random selection. This strategy is more sophisticated than the previous one and provides the selecting agent with a way to choose other agents according to possible past interactions. In our scenario, these interactions are given by the history of past interactions that the *Civilian Protection Unit* stores.
— GEA Strategy: this strategy takes into account both the trust and the matching scores (as described in Section 5). In the GEA selection strategy, we choose to com-

pute the GEA score following Algorithm 2 with $v = 0.5$. As before, the agent with the highest GEA score is selected for each location, and if more than one agent has the same highest GEA score, one is selected randomly.

7.2. Experiment Setup

In order to test the selected scenario we have built a system in which we can control several variables relevant for testing the selection strategies discussed in the previous section. These variables are:

(1) The *correctness* of the water level readings. *Correct* agents always return accurate water level reading, and *incorrect* agents always return water level readings with some error.
(2) The *matching score* defines how similar the function of a *sensor* agent is with respect to what the *Civilian Protection Unit* expects. For example, a *sensor* agent can provide a service to read the water level with one more parameter than what the *Civilian Protection Unit* expects.
(3) The *format* of the results returned by the agent. This variable tests the cases in which an agent, even though it correctly returns the results (no error added) and perfectly matches expected function signature, returns the results in a format different from what it was expected, e.g. in inches instead of meters.

By considering these variables we define the following types of agents:

— **Perfect *sensor* agent:** an agent with correct behavior (therefore not adding any error to the water-level), a perfect matching score, and a format that is expected. This is the ideal case where the agent will give correct results.
— **Good *sensor* agent:** an agent with a correct behavior, a matching score lower than 1, and a format that is expected. This is the case where the agent will give correct results, even though it does not have a perfect matching score.
— **Damaged *sensor* agent:** an agent with incorrect behavior (e.g., because it's damaged), perfect matching score, and a format that is expected. In this case the agent will be giving inaccurate results.
— **Foreigner *sensor* agent:** an agent with correct behavior, but with a format that is unexpected. This case can be understood as if the agent has a correct behavior (the water-level value is correct), but the agent *"speaks a different language"*, resulting in an overall inaccurate water-level value (in the sense the value is not what it was expected).
— **Damaged foreigner *sensor* agent:** an agent with incorrect behavior, and that *"speaks a different language"* too. Then the resulting value will be inaccurate.

7.3. Results

Several configurations were defined for the experiments in a way that for each configuration, at least one perfect or good sensor agent was present. The rationale of the experiment was to test the capability of each of the selection strategies (introduced earlier by Section 7.1) to correctly and continuously identify the most accurate agents to interact with; and how the selected variables (introduced earlier by Section 7.2) affected the performance of the Selection Strategies. Each experiment configuration ran 15 times in order to get statistically significant results. This document presents only the most relevant results, while we refer the reader to our technical document [Pane et al. 2008] for more comprehensive results.

In order to evaluate the results of a given selection strategy, we defined a quality metric. We used the *mean absolute error* metric defined in Equation 18. This error measures how accurate a given selection strategy is in selecting agents with correct

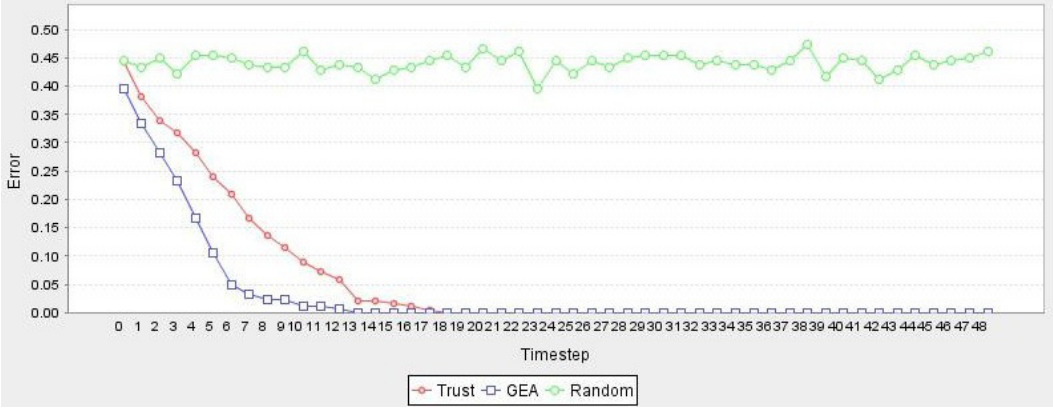

Fig. 5. Mean absolute error for Experiment A. (6 locations, 20 agents per location, each with one perfect agent)

results. We can compute this error given that we know the expected water-level of each location for each timestep.

$$\mathrm{error_t} = \frac{1}{n} \cdot \left(\sum_{i=1}^{n} |e_i| \right), \tag{18}$$

where t denotes the time step, n denotes the number of locations, and e_i is the error of each water level reading as defined in Equation 19. In Equation 19, $water\_level_o$ denotes the obtained water-level (i.e. the result given by a *sensor* agent), and $water\_level_r$ denotes the real water-level, as simulated in our experiment.

$$\mathrm{e_i} = \begin{cases} 0.5 & \text{if} (water\_level_o \neq water\_level_r) \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

Note that the error of each reading (e_i) is not defined as a difference between the real level and the claimed one ($(water\_level_o - water\_level_r)$) as intuitively one might expect. This is because we wanted to evaluate how many correct agents were selected by the selection strategy, and not the overall difference between the real water-level value $water\_level_r$ and the claimed water-level value $water\_level_o$.

Figure 5 shows the results for Experiment A, where 120 agents were generated for 6 locations (20 agents per location) and where each location contained at least one *perfect agent* (see Section 7.2). The experiment ran for 50 timesteps. The results show that selection strategies that are based on the use of information learned in previous interactions improves agent selection. As we can see, the Trust based selection strategy is clearly better than the Random selection strategy. In this experiment configuration, GEA clearly outperforms the other two selection strategies. This means that in a setting where we have at least one perfect agent per location, the use of the additional information provided by the matching score in GEA contributes to a faster convergence of the selection strategy.

Figure 6 shows the results for Experiment B. In the definition of this experiment, we wanted to see the effect of not having perfect agents in the locations. Therefore, we generated 120 agents also for 6 locations, but in this case, each location contained only a *good agent*. The experiment was also run for 50 timesteps. In this experiment there are no agents with both perfect matching score and correct behavior (*perfect agent*), but we do have incorrect behaving agents with perfect matching (*damaged sensor agents*).

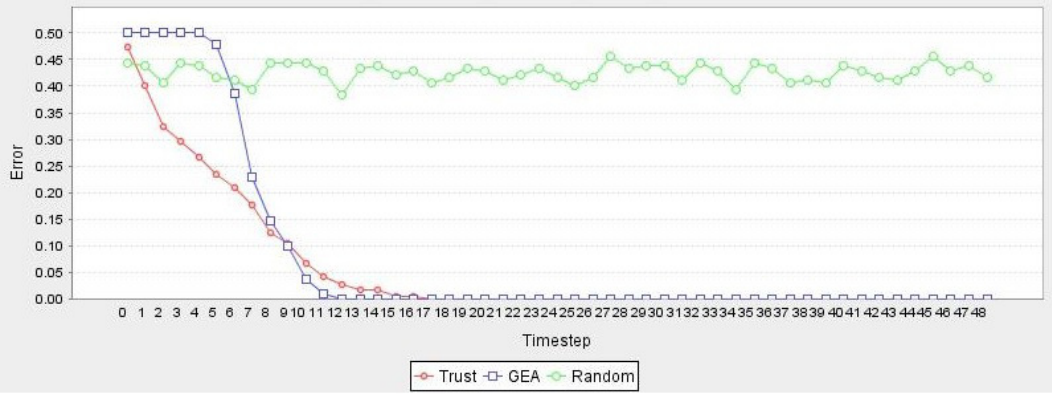

Fig. 6. Mean absolute error for Experiment B. (6 locations, 20 agents per location, no perfect agents)

The experiment is meant to stress test the selection strategies and to study how the presence of such *damaged sensor agent* affects their performance. The question we want to answer is how much would it take (in terms of timesteps) a selection strategy to realize the agents with perfect matching scores are giving incorrect results.

As we can see from Figure 6, the Random and Trust selection strategies maintain the same behavior as in Figure 5, while the GEA selection strategy clearly changes the mean absolute error in the initial time steps. This change can be understood if we take into account that GEA uses matching information in order to select the agents. This means that, when there is no previous information (previous interactions with the agents), GEA will try to select first those agents having higher matching score: that is all the bad agents (recall that in this experiment the agents with a perfect matching score have an incorrect behavior). However, during the first time steps GEA learns (via Trust) how the selected agents behave incorrectly and, as a consequence, starts to select agents from the second best matching score group. After some time steps, GEA finds the correct behaving agents and starts to select them, giving results as good as the ones returned by Trust; finally in subsequent time steps, GEA even outperforms Trust. This highlights a potential drawback of using matching - and therefore GEA - information: a perfect matching score does not guarantee a good performance, because even though the requirements of the role were well understood and interpreted, the process used to produce the results were faulty. However, this is a potential problem with *any* agent, and we would in general expect an agent with a better approximation to the required abilities to produce a better outcome. This experiment represents a worst-case scenario, in which the agents with the best matching scores in fact produce the worst results, and in which no trust information is available about any of the agents prior to interaction, but must be gathered as the interactions proceed. This allows us to demonstrate the strength of GEA: even in such a worst-case scenario, the combination of the matching and trust means that the algorithm is quickly able to identify damaged or erratic agents and filter them, leading to results better than using trust scores alone within only a few time steps. In this case, using trust alone is an advantage because it is initially picking agents randomly. Experiments A and B therefore demonstrate that GEA gives better results than trust scores alone if and only if the agents with a higher matching score do, in fact, perform better. When this is not the case, GEA does still appear to ultimately perform better than trust, but will perform worse in the first few time steps, as it is learning to disregard the misleading matching information.

Experiment B also provides interesting insight into how GEA would perform with dishonest agents. Although none of the agents in the experiment were dishonest, the behaviour of an agent with good matching but a broken method of producing results is the same as an agent with poor matching who is pretending to have a high matching score: the agent performs much worse than its reported matching score would lead one to suspect. Since GEA was able to identify and filter out such agents fairly quickly, this experiment demonstrates our claim in Section sec:ex that it is not possible to game this system long-term through lying.

Figures 5 and 6 illustrate that the number of time steps needed for a given selection strategy to converge to an average error of 0 is dependent on the number of agents they have to select (20 in these cases). This observation might be intuitive, since, in the worst case, a selection strategy based on the use of past information could first select all the agents giving incorrect results, and then select agents giving accurate results.

We have also tested other scenarios with 4, 10 and 40 agents per location, and in all the cases, Trust and GEA selection strategies comply with the worst case expected results (i.e. the error is minimized to 0 in a time step that is less than or equal to the total number of agents per location). The general shape of the curves remains similar. We have also created experiments increasing the number of locations but in these experiments the general shape of the mean absolute error curves did not change. In other words, an increase in the number of locations did not affect the general behaviour of the selection strategies. The Random selection strategy had, in all the cases, an average error of 0.5, as expected. For additional information and more detailed results, we refer the interested reader to [Pane et al. 2008]. Experiments over a different use case has also been presented by [de Pinninck et al. 2008].

Another consideration is the amount of resources that are necessary to run the different algorithms. We measured this in terms of the amount of time that each interaction took to set up and run. There are several stages of this process: locating the appropriate IM and retrieving the list of agents subscribed to each role; allowing those agents to provide their matching scores; allowing those agents to decide which other agents they wish to interact with (using one of the three strategies: choosing randomly, calculating Trust scores, and calculating GEA scores); ordering the allocation of roles on this basis; executing the IM with the selected peers. Since only one stage varies as the selection mechanism varies (the stage in which the agents choose whom they wish to interact with), we can use the overall runtime as a way of measuring the resources this step requires for each of the selection strategies. Average results are shown in Figure 7.[15]

As expected, the times for random selection remain constant across all timesteps, and approximately estimate the amount of time required for all the stages other than that of agent selection (because this stage takes close to zero time when agents are selected randomly). However, both Trust and GEA required more computational time to complete the interaction. The improvement in the selection of the peers gained using Trust or GEA has a significant computational cost, that goes from a few seconds in the initial time steps to an increase of almost 30s for Trust and 40s for GEA in the final time steps. The constant increase of the computational time along the time steps is due to the fact that, at each time step, the Trust selection strategy uses all the information about the previous interactions to compute the trust score, thus increasing the amount of computation needed in each time step. This increase in time may not be acceptable in highly dynamic environments; on the other hand, the increased quality

[15]Note that these times are for an experiment with slightly different parameters to the ones discussed in this paper; however, we do not expect that this will make any difference to the trend observed.

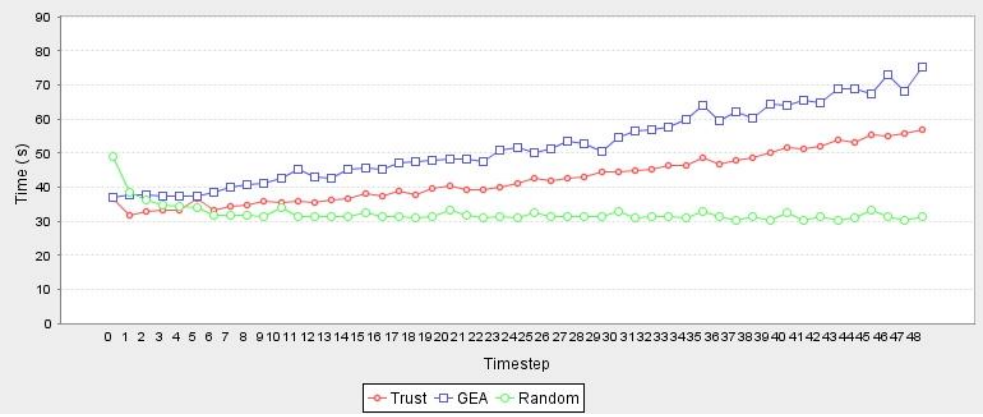

Fig. 7. Time needed for setting up and running the interactions

of the interaction may outweigh the extra time this takes. Such judgements depend on the context in which the interaction is taking place.

8. RELATED WORK

Different approaches may be used for finding suitable agents to interact with. Examples of these are formal verification techniques such as model checking for verifying the suitability of potential collaborators (e.g. [Osman and Robertson 2007]), matching algorithms for job recruitment in the real world (e.g. [Bizer et al. 2005]), and so on. In this paper, however, we tackle the problem from a different angle by calculating a socio-cognitive trust measure that is based on the agent's performance in similar past experiences.

In the context of trust, however, we should clarify that the term 'trust' is a term that has been used by different communities to refer to slightly different notions. Ramchurn et al. [2004] provides an interesting overview of those notions that vary from individual level trust, where agents may form beliefs about the trustworthiness of others, to system level trust, where security protocols ensure the trustworthiness of interactions. According to their categorisation, the proposed trust model in this paper is considered on the individual level, and it follows socio-cognitive approaches by basing the notion of trust on the capability and willingness of agents, as well as experience based approaches, since trust measures are based on past experiences.

The proposed trust model calculates the expectation about an agent's performance in a given context by assessing the agent's willingness and capability through the semantic comparison of the current context with the agent's history of experiences. The distinction between willingness and capabilities has been proposed earlier by Castelfranchi and Falcone [1998; 2000]. Several other approaches, such as those of Şensoy and Yolum [2006], Teacy et al. [2006], and Rehak et al. [2006], have used a 'contextual' approach when dealing with trust, which is also the central theme of this paper.

According to the classification proposed by Sabater and Sierra [2005] the trust model proposed in this paper is based on direct experiences, is subjective (the trust value, although might be shared, is local to each individual), is context dependent, it does not provide means for the exchange of information among agents (e.g. gossiping) but it does accommodate such exchange, it assumes that the agents may lie, and it incorporates a reliability measure.

The proposed model is in coherence with the theory of Castelfranchi and Falcone [2001] that states that (1) only a cognitive agent endowed with goals and beliefs can trust another agent, (2) trust is a mental state describing one agent's attitude towards another with respect to a given behaviour/action for achieving some goal, and (3) trust is the mental counter part of delegation. In our model, delegation is defined through commitments, and trust is then held by one agent about another, assessing its behaviour/action in future commitments. The agent assessing the other holds beliefs about what its goals are and how the other agent behaved in the past. However, we do not discuss how commitments are brought about (whether through argumentation or other means). In other words, we do not address Castelfranchi and Falcone [2001]'s dependence belief which states that the trusting agent believes it needs the trusted agent and hence delegates its task to it. Furthermore, we also do not address the motivation belief which states that it is believed that the trusted agent has motives for fulfilling its commitments as our work does not address the issue of motives. What we do consider, however, are: (1) the competence belief which states that the trusted agent has the capability of fulfilling its commitments, and we address this issue in Section 3.2.1; (2) the disposition/willingness and persistence belief which states that it is believed that the trusted agent has the willingness and persistence to fulfil its commitments, and we address this issue in Section 3.2.2; and (3) the self confidence belief which states that the trusted agent has self confidence that it can perform the action it has been trusted with, and we address this issue through the matching algorithm of Section 4.

From the large existing literature on trust, the model that can be classified closer to ours is the early model by Marsh [1994]. Differently from ours, Marsh's model follows a utilitarian approach and time decay is modelled as a time window for experiences. The model by Abdul-Rahman and Hailes [2000] uses a qualitative degree approach to model trust and takes into account the context as well. However the modelling of uncertainty is somewhat ad-hoc and not based on probabilistic grounds. The REGRET model [Sabater and Sierra 2002] has some similarities in the time decay consideration and on the subjective modelling of the experiences. However, the overall notion of trust does not have a probabilistic meaning and is based on a utility modelling of the interactions that we depart from (see Debenham and Sierra [2009] for a discussion). AFRAS [Carbo et al. 2003] offers a model based on fuzzy sets with a similar, entropic-like, notion of uncertainty on the behaviour of other agents. Ramchurn et al. [2004] also based reputation on similarity between new contexts and past ones. However, their approach uses the concept of fuzzy sets to compute one's confidence, based on the notion of assigning utilities to the different aspects of a context. Trust is then built on the concept of the maximum expected loss in utility. Sierra and Debenham [2006] distinguished between trust, which measures the expected deviations of behaviour in the execution of commitments, and honour, which measures the expected integrity of the arguments exchanged. They also distinguished between capabilities and trust, whereas our trust approach incorporates the notion of capabilities.

Sierra and Debenham [2007] uses the same philosophy as this paper, and some general equations have first been introduced there. However, there are several distinctions to be made. This paper: (1) introduces the distinction between capability and willingness (Equation 1); (2) illustrates how trust may be combined with self confidence (i.e. the matching score) to result in a better selection mechanism; (3) elaborates further on the notion of ontology; (4) provides a richer description of context (in terms of interaction model, the roles, the messages, the messages committed to, and the constraints), resulting in richer descriptions of commitments, observation, and experiences, as well as richer similarity measures; (5) provides a new approach for aggregating trust (Equa-

tion 15); and (6) presents a sample trust algorithm, which has not been presented before.

Simari et al. [2008]'s approach is also similar to ours in the sense that it compares what the agent has committed to to what is actually delivered (which we refer to as observed). Similar to us, trust is then context dependent and based on past performances, and we both account for partial fulfillments or fulfillments of variants of commitments, although we achieve that through our similarity measures and breaking commitments into a conjunctive set (disjunctive sets may easily be accommodated for, as it has been illustrated by Section 3.1.5). However, to account for late commitments, we require time to be defined as part of the commitment. Unlike Simari et al. [2008], our similarity measures are based on semantic matching. Furthermore, we not only define a comparison with past performance to compute trust, but we use it to distinguish capabilities from willingness. We also introduce the notion of information decay, which accounts for change in behaviour over time.

Schillo et al. [2000]'s approach does not consider dependence on the context. Their approach focuses on the trustworthiness of the source of a gossip, which we do not focus on. As such, at first sight, their proposed approach may be thought of as complementary to ours as it may be used for calculating the reliability of the source providing a trust measure (this has been addressed in Section 3.1.4 and the reliability measure $R^t(\alpha, \beta, \mu)$ was referred to in Section 3.2.2). However, we note that Schillo et al. [2000]'s approach assumes that the details of past commitments and their observations are not communicated during gossip, making it impossible to pinpoint correlated gossip. Our approach, on the other hand, assumes that the details of past observations are communicated during gossip.

Lastly, Dondio and Barrett [2007]'s approach is a more general approach that tries to understand the application domain and the structure of the trust model in order to match trust models with application domains. Their model may be adapted to incorporate ours as one example of their trust schemes.

9. CONCLUSION

This paper addressed the problem of finding suitable collaborators in open distributed systems. When choosing an agent to interact with, one needs the agent to declare how well it can perform its role. This represents the self confidence of the agent in question, as described by Castelfranchi and Falcone [2001]. For this, this paper suggests a matching algorithm that would match the agent's capabilities to those it is planning to commit to. Nevertheless, agents cannot always be trusted. They may be malicious, misinformed, suffer from miscommunication, and so on. As such, there is also a need to combine this matching score with a score describing how much trusted is the agent in performing the given role. For this, a novel trust model has been proposed, which is the main contribution of this document. The novelty of the proposed trust model is that it calculates the expectation about an agent's future performance in a given context by assessing both the agent's willingness and capability (as suggested by Castelfranchi and Falcone [2001]) through the semantic comparison of the current context in question with the agent's performance in past similar experiences.

The proposed model is generic enough to be applied to a variety of scenarios. It may compute the trust in both human agents or software agents. In short, this paper essentially provides a mechanism for assessing trust that may be applied to any real world application where past commitments are recorded and observations are made that assess these commitments. In such scenarios, the proposed model can be used to calculate one's trust in another with respect to a future commitment by assessing the other's past performance. Other than the e-response flood scenario presented in this paper, this work has already been applied to the field of supplier rela-

tionship management by evaluating past orders to support future supplier selection (http://www.iiia.csic.es/SRM/) [Fabregues et al. 2009]. It is also currently being applied to the *incondicionales.com* football forum (a member of the international *fanscup.com* football social network). In the case of *incondicionales.com*, the proposed trust model is used to help assess the trust in a forum member (whether s/he was a normal user, a moderator, an administrator, and so on) based on how well they have been complying to previous commitments, where commitments are defined by the forum's rules and regulations.

Acknowledgements

This work is supported by the OpenKnowledge project (funded by the EU under grant FP6-027253), Agrement Technologies project (funded by the Spanish ministry of science and innovation under grant CONSOLIDER CSD2007-0022, INGENIO 2010), ACE project (recommended for national funding by the EU CHIST-ERA program), and CBIT project (funded by the Spanish ministry of science and innovation under grant TIN2010-16306).

REFERENCES

ABDUL-RAHMAN, A. AND HAILES, S. 2000. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6 - Volume 6*. HICSS '00. IEEE Computer Society, Washington, DC, USA, 6007–.

ARCOS, J. L., ESTEVA, M., NORIEGA, P., RODRÍGUEZ-AGUILAR, J. A., AND SIERRA, C. 2005. Engineering open environments with electronic institutions. *Engineering Applications of Artificial Intelligence Journal 18*, 191–204.

BIZER, C., HEESE, R., MOCHOL, M., OLDAKOWSKI, R., TOLKSDORF, R., AND ECKSTEIN, R. 2005. The impact of semantic web technologies on job recruitment processes. In *Wirtschaftsinformatik 2005*, O. K. Ferstl, E. J. Sinz, S. Eckert, and T. Isselhorst, Eds. Physica-Verlag HD, Berlin, 1367–1381.

CARBO, J., MOLINA, J., AND DAVILA, J. 2003. Trust management through fuzzy reputation. *International Journal of Cooperative Information Systems 12,* 1, 135–155.

CASTELFRANCHI, C. AND FALCONE, R. 1998. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems*. IEEE Computer Society, Washington, DC, USA, 72.

CASTELFRANCHI, C. AND FALCONE, R. 2000. Trust is much more than subjective probability: Mental components and sources of trust. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*. IEEE Computer Society, Washington, DC, USA, 6008.

CASTELFRANCHI, C. AND FALCONE, R. 2001. Social trust: A cognitive approach. *Trust and deception in virtual societies 2005,* 11 July, 55–90.

ŞENSOY, M. AND YOLUM, P. 2006. A context-aware approach for service selection using ontologies. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, New York, NY, USA, 931–938.

DE PINNINCK, A. P., SIERRA, C., WALTON, C., DE LA CRUZ, D., ROBERTSON, D., GERLOFF, D., JAEN, E., LI, Q., SHARMAN, J., ABIAN, J., SCHORLEMMER, M., BESANA, P., WAI LEUNG, S., , AND QUAN, X. 2008. Summative report on bioinformatics case studies. Project Deliverable D6.4, The OpenKnowledge project. URL: `http://www.cisa.inf.ed.ac.uk/OK/Deliverables/D6.4.pdf`.

DEBENHAM, J. AND SIERRA, C. 2009. An agent supports constructivist and ecological rationality. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, R. Baeza-Yates, J. Lang, S. Mitra, and Simon, Eds. WI-IAT '09. IEEE Computer Society, Washington, DC, USA, 255–258.

DONDIO, P. AND BARRETT, S. 2007. Presumptive selection of trust evidence. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. AAMAS '07. ACM, New York, NY, USA, 166:1–166:8.

FABREGUES, A., MADRENAS-CIURANA, J., SIERRA, C., AND DEBENHAM, J. 2009. Supplier performance in a digital ecosystem. In *Proceedings of the IEEE International Conference on Digital Ecosystems and Technologies (IEEE-DEST 2009)*. 466–471.

GIUNCHIGLIA, F., MCNEILL, F., YATSKEVICH, M., PANE, J., BESANA, P., AND SHVAIKO, P. 2008. Approximate structure-preserving semantic matching. In *7th International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2008)*. Springer, Berlin, Heidelberg.

GIUNCHIGLIA, F. AND SHVAIKO, P. 2003. Semantic matching. *Knowledge Engineering Review 18,* 3, 265–280.

GIUNCHIGLIA, F., SIERRA, C., MCNEILL, F., OSMAN, N., AND SIEBES, R. 2008. Deliverable 4.5: Good enough answers algorithm. Tech. rep., OpenKnowledge Project.

GIUNCHIGLIA, F. AND WALSH, T. 1992. A theory of abstraction. *Artificial Intelligence 57*, 323–389.

LAMANNA, D. D., SKENE, J., AND EMMERICH, W. 2003. Slang: A language for defining service level agreements. In *FTDCS '03: Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*. IEEE Computer Society, Washington, DC, USA, 100.

LI, Y., BANDAR, Z. A., AND MCLEAN, D. 2003. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Trans. on Knowl. and Data Eng. 15*, 871–882.

MARSH, S. 1994. Formalising trust as a computational concept. Ph.D. thesis, Department of Mathematics and Computer Science, University of Stirling.

OSMAN, N. AND ROBERTSON, D. 2007. Dynamic verification of trust in distributed open systems. In *Proceedings of the 20th international joint conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1440–1445.

PANE, J., SIERRA, C., TRECARICHI, G., MARCHESE, M., BESANA, P., AND MCNEILL, F. 2008. Summative report on gea, trust and reputation: Integration and evaluation results. Project Deliverable D4.9, The OpenKnowledge project. URL: `http://www.cisa.inf.ed.ac.uk/OK/Deliverables/D4.9.pdf`.

PINYOL, I., SABATER-MIR, J., AND CUNÍ, G. 2007. How to talk about reputation using a common ontology: From definition to implementation. In *Proceedings of the Ninth Workshop on Trust in Agent Societies. Hawaii, USA*. 90—101.

Pitt 2007. The Pitt review: Lessons learned from the 2007 floods. HMSO, London.

RAMCHURN, S., SIERRA, C., GODO, L., AND JENNINGS, N. R. 2004. Devising a trust model for multi-agent interactions using confidence and reputation. *International Journal of Applied Artificial Intelligence 18,* 9-10, 833–852.

RAMCHURN, S. D., HUYNH, D., AND JENNINGS, N. R. 2004. Trust in multi-agent systems. *Knowl. Eng. Rev. 19*, 1–25.

REHAK, M., GREGOR, M., PECHOUCEK, M., AND BRADSHAW, J. M. 2006. Representing context for multiagent trust modeling. In *IAT '06: Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*. IEEE Computer Society, Washington, DC, USA, 737–746.

ROBERTSON, D. 2005. A lightweight coordination calculus for agent systems. In *Declarative Agent Languages and Technologies II*, J. A. Leite, A. Omicini, P. Torroni, and P. Yolum, Eds. Lecture Notes in Computer Science Series, vol. 3476. Springer, Berlin, Heidelberg, 183–197.

RUBNER, Y., TOMASI, C., AND GUIBAS, L. J. 1998. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*. ICCV '98. IEEE Computer Society, Washington, DC, USA, 59–.

SABATER, J. AND SIERRA, C. 2002. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. AAMAS '02. ACM, New York, NY, USA, 475–482.

SABATER, J. AND SIERRA, C. 2005. Review on computational trust and reputation models. *Artificial Intelligence Review 24,* 1, 33–60.

SCHILLO, M., FUNK, P., AND ROVATSOS, M. 2000. Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence 14,* 8, 825–848.

SIERRA, C. AND DEBENHAM, J. 2006. Trust and honour in information-based agency. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. AAMAS '06. ACM, New York, NY, USA, 1225–1232.

SIERRA, C. AND DEBENHAM, J. K. 2007. Information-based agency. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, M. M. Veloso, Ed. 1513–1518.

SIMARI, G. I., BROECHELER, M., SUBRAHMANIAN, V. S., AND KRAUS, S. 2008. Promises kept, promises broken: An axiomatic and quantitative treatment of fulfillment. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference*, G. Brewka and J. Lang, Eds. AAAI Press, 59–69.

TAI, K.-C. 1979. The tree-to-tree correction problem. *Journal of the ACM 26*, 422–433.

TEACY, W. T. L., PATEL, J., JENNINGS, N. R., AND LUCK, M. 2006. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems 12,* 2, 183–198.

Received February 2012; revised ; accepted