OpenKnowledge

FP6-027253

Good Enough Answer Algorithms

Fausto Giunchiglia[1], Carles Sierra[2], Fiona McNeill[3], Nardine Osman[3], Ronny Siebes[4]

[1] University of Trento
[2] IIIA(CSIC) Barcelona
[3] University of Edinburgh
[4] Vrije Universiteit Amsterdam

Report Version: final
Report Preparation Date: 31/08/07
Classification: deliverable D4.5
Contract Start Date: 1.1.2006 Duration: 36 months
Project Co-ordinator: University of Edinburgh (David Robertson)
Partners: IIIA(CSIC) Barcelona
Vrije Universiteit Amsterdam
University of Edinburgh
KMI, Open University
University of Southampton
University of Trento

1 Introduction

This deliverable outlines the process through which a peer, or a peer's user, may determine whether a potential interaction is good enough to justify attempting to execute it. This judgement involves both discovering an appropriate interaction model and, which is more difficult, determining whether other peers who may be involved in that interaction are likely to perform their roles to a level which is judged good enough. Peers who are judged unlikely to reach this standard will be rejected as potential interaction partners.

In determining whether peers will be likely to perform acceptable, there are two main factors to consider:

- **How well the peer claims to be able to perform this role and what restrictions it places on the role.** These factors can be determined by the matching process: any peer performing a role will match its abilities to the constraints of that role to produce a numerical score that determines how close this match is and qualitative matchings that describe the areas where this matching is only approximate. Ideally, the peer will only commit to actions that are within its own abilities (though malicious peers may commit to actions outside the scope of their abilities, making trust measures crucial); therefore, its commitments are not to the actions described in the role but to its mappings of these to its own abilities. The matching score is a summarisation of how close a peer's abilities are to those required of the role.

  This process is internal to the peer and the results cannot be externally verified by other peers, who will not have full access to that peer's knowledge base, so it is possible for mistakes and deception to occur.

- **How well trusted that peer is.** Measuring trust is complex because it is usually context-dependent. A peer that has performed well for a particular role in the past will be trusted more to perform the same role again, or a similar role, than it will to perform a completely different role. Additionally, since, in a large peer-to-peer system, peers will often wish to interact with other peers with whom they have no interaction history, it is useful for peers to have means other than personal experience for assessing trust. This could be through the sharing of experiences and opinions about peers' abilities by peers on the network.

The format of this document is as follows. Section 2 explains in detail the trust measures that are used and the different aspects that need to be taken into consideration when determining trust values. Section 3 discusses the integration of the trust and matching measurements to enable peers to determine whether other peers are likely to be good enough at performing their intended roles. Note that we do not explain the details of our matching process in this deliverable, as these have already been discussed in detail in Deliverable 4.4. Section 4 outlines a simple interaction lifecycle to explain how the issues described in Sections 2 and 3 are pertinent within the OpenKnowledge context and to highlight where 'good enough' choices need to be made and where matching and trust measurements are necessary and how they are used. Section 5 summarises the document and discusses how we intend to follow on from this work to present a complete solution to good enough answers in OpenKnowledge.

2 Trust Model

2.1 Preliminaries

Our basic tenet on trust is that it may be defined as a cognitive state that a person α holds with respect to the expected behaviour of another person β on some matter φ. We assume that each peer has a fixed[1] local ontology O and the φ is a correct term from the set of terms built on top of O, denoted as $Term(O)$.[2] This 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: $P(Observe(\alpha, \beta, \varphi)|Commit(\beta, \alpha, \varphi))$, that is, the probability of α observing φ given that β made a commitment to α to perform φ. In P2P systems, persons become peers and in multiagent systems persons become agents. In general, the holders and subjects of trust are programs.

This section is about how to estimate $P(Observe(\alpha, \varphi)|Commit(\beta, \alpha, \varphi))$. We consider that the observation of φ is a direct consequence of the capability of agent β to perform φ and the willingness of agent β to do so. That is, $P(Observe(\alpha, \varphi)|Commit(\beta, \alpha, \varphi))$ becomes $P(Can(\beta, \varphi) \textit{ and } Does(\beta, \varphi)|Commit(\beta, \alpha, \varphi))$ which is equal to $P(Can(\beta, \varphi)|Commit(\beta, \alpha, \varphi)) \cdot P(Does(\beta, \varphi)|Commit(\beta, \alpha, \varphi))$ and because capabilities are not related to commitments we have

$$\begin{aligned} P(Observe(\alpha, \varphi)|Commit(\beta, \alpha, \varphi)) = \\ P(Can(\beta, \varphi)) \cdot P(Does(\beta, \varphi)|Commit(\beta, \alpha, \varphi)) \end{aligned} \tag{1}$$

We will estimate $P(Does(\beta, \varphi)|Commit(\beta, \alpha, \varphi))$ based on past observed behaviour in similar circumstances (similar commitments) and $P(Can(\beta, \varphi))$ as a matching degree between the capabilities needed now and the capabilities observed in the past. Then, trust will become an entropy measure of the resulting probability distribution $P(Observe(\alpha, \varphi)|Commit(\beta, \alpha, \varphi))$. We will simplify notation in the rest of the paper and will refer to the previous probability distributions as $P(\varphi \mid \varphi)$.

Interaction models Interaction models can be atomic, that is, they code a particular behaviour protocol, or can be combinations of atomic ones to represent more complex activities, usually distributed along time. Thus, we assume that IMs can be combined following a simple grammar[3]:

$$IM ::= im \mid IM \cdot IM \mid IM \mid\mid IM \mid IM^*$$
$$im ::= \texttt{see LCC manuals for details}$$

[1]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.

[2]in LCC interaction models the content of a message can be a Cartesian product of terms instead of a single term. All the ideas in this paper can be naturally extended to this particular case.

[3]this grammar is a simplification of the expressive possibilities of performative structures in electronic institutions [1], where also the flow of agents and roles is specified.

where the $\cdot$ represents sequence, $||$ is the concurrent execution of two IMs and $*$ iteration. We will note in capital letters: IM, when an interaction model can be atomic or a combination of atomic ones, and by low case letters: im, when it is an atomic interaction model.

Local ontologies We assume each peer has a local ontology O that consists of some basic (and finite) data types T with two relationships: refinement and combination. A refinement relationship (named usually as *is-a*) $: \ \subseteq T \times T$ defines increased specificity (e.g. a relation between values and types). The combination relationship (named usually as *part-of*) $\subset \ \subseteq T \times T$ defines the components of an entity. We represent this relation as a term where the predicate is the composed entity and the arguments the parts of it (e.g. car(wheels, engine) means wheels $\subset$ car and engine $\subset$ car). We can naturally extend these relationships to tuples of values and also to terms built from a function symbol and a set of arguments:

- If $t'_i : t_i$ then $t_1 \times ... \times t'_i \times ... \times t_n : t_1 \times ... \times t_i \times ... \times t_n$
- If $p_1 \subset f, ..., p_n \subset f$ and $g : f$ then $g(p_1, ..., p_n) : f(p_1, ..., p_n)$, also if $(p'_i, ..., p'_n) : (p_i, ..., p_n)$ then $f(p'_i, ..., p'_n) : f(p_i, ..., p_n))$.

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

The : relationship over the set of terms defines a tree relationship (in fact, a forest with different roots that are not comparable to each other). The more specific the term, the lower in the tree. That is, 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. This tree, or trees, can be generated on the fly and locally to a particular term, that is all ancestors, siblings and descendants around the term and for a certain distance of the term so as to focus on a particular region of the terminology. This is specially important to obtain trust values in an efficient way.

Commitments In the context of OpenKnowledge, the behaviour of agents is somehow constrained to the execution of interaction models (IM). Any commitments that an agent might create (e.g. I'll deliver ten bottles of wine tomorrow) are direct consequences of the execution of interaction models. In fact they are constraints on the future behaviour of the peer in the execution of an IM. Commitments come in two forms:

- *Norms*, representing restrictions on the peer's behaviour due to the simple fact that a peer accepts to play a role in an interaction model. That is, the agent commits to speak when it is time to speak, use the right messages, and in general to follow the script of the IM and to satisfy its constraints in the precise manner defined by the IM.
- *Agreements*, as direct consequences of the choices that an agent makes when playing an interaction model and that will affect its future behaviour. For instance, we

expect that when executing the IM for the delivery of the bottles —after having bought 10 Château Rotschild, the variable in the message corresponding to the number of bottles gets instantiated to 10. Since we allow approximate matching, it may be that the peer cannot satisfy the constraints as defined in the IM but instead will match the constraints to its own abilities. A set c_m is built up for every message in an IM that details the constraints the peer will satisfy before passing the message and in order to instantiate the variables in the message. In the case of perfect matching, c_m will be equivalent to the set of constraints on the message as described in the IM. However, in the case of approximate matching, at least one of these constraints will differ. It is made clear in c_m exactly which original constraints altered constraints are mapped to.

We represent both types of commitments (Norms and agreements) as

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

where β commits to α (another peer) to play role r in (atomic) interaction model im and, as part of playing that role, to send a particular message m instantiated as φ. φ is determined through the matching of constraints detailed in c_m, the commitment being made at time t. That means that α expects β to later on execute im playing role r and that β will instantiate message m as φ, that is $\varphi : m$ in a manner determined by the set of constraints c_m. If c_m is not identical to the constraints on m in im, the expectations of instantiations of the variables may differ. The trust that α will build on β will be determined by how β keeps its commitments.

Given any message, we want to be able to automatically determine whether that message was passed correctly during the running of an im. We can be sure that the format of the message is correct: passing messages that are structurally incorrect with respect to the im is impossible in the system. It therefore needs to be determined whether the variables in the message have been instantiated correctly. The original constraints in the im describe how the variables of m were expected to be instantiated through giving type information. φ, which is derived from c_m, describes how the peer is intended to satisfy them which, in the case of approximate matching, will not be the same. For example, a message in im may be $ask(C, Ma, Mo)$ and these variables may be instantiated through the satisfaction of the constraint $need(C : car, Ma : make, Mo : model)$ on that message. In the case of perfect matching, the receiving peer would expect C to be some kind of car, Ma to correspond to make and Mo to model, as detailed in the constraint and therefore we have $\varphi = ask(C : car, Ma : make, Mo : model)$. However, if $c_m = \{need(C : vehicle, Ma : make, Mo : \bot)\}$ then the peer who created c_m will generate $\varphi = ask(C : vehicle, Ma : make, Mo : \bot)$. Note that φ contains type information where m does not because this type information has been derived from the constraints.

Given any constraint over a message, it can be automatically estimated whether the constraint was satisfied during the running of an im. That is, we assume the existence of a function $g : MSG \times OBS \mapsto \{\top, \bot\}$ that reports for each execution of an im, whether a certain φ has been satisfied when generating a message $m \in MSG$ according to an observation $\varphi' \in OBS$ by returning $\top$ or if it was violated or the im never finished by returning $\bot$.

After a commitment is made by agent β, agent α can then observe the execution of the commitment and giving a value in a certain evaluation space D. For instance, a numerical scale $[0, 1]$ could be used for automatic determination of constraint violation, determined by g, and a qualitative set $\{bad, neutral, good, very\_good\}$ could be used for user feedback. However, in general this user feedback would not be for a specific message but rather for the interaction as a whole. This interaction-specific feedback would then be propagated to the messages, so that the performance of any message is assumed to be of the same value as the performance of the whole im.

The execution of an im will possibly generate new commitments for future behaviour in ulterior ims. For example, we agree on the product to buy and the price and at the same time commit on some conditions for the ulterior negotiation of the payment method: I'll not charge any bank commissions if you decide to pay by credit card. In this way we can naturally think of an im played by an agent as an 'operation' that consumes commitment(s)[4] and generates commitments. That leads to our basic representation unit hold by any agent α about the behaviour of other agents, which is:

$$\begin{aligned}\mu =&\langle \beta, IM,\\ &\{\langle Commit(\beta, \alpha, \langle im, r, m, \varphi, c_m\rangle, t), \varphi', g(\varphi, \varphi'), d\rangle\}_{im\in atom(IM), m\in msg(im)}, C\rangle\end{aligned}$$

where β engages in the execution of im and C is the set of new commitments generated during the execution of im affecting α. Each element of the set records the performance of β for a particular message m for which there was a constraint $\varphi : m$ with constraint commitments c_m, and for which α had observation φ', evaluated performance by $d \in D$ and noted termination/satisfaction by $g(\varphi, \varphi')$.

We denote the set of all existing μs as M. In the rest of the document we will often view any of the recorded experiences from the perspective of a single message commitment and will abuse notation writing $\mu = (\varphi', \varphi)$. We will note the data base of experiences of α as M_α

Reputation Gossipping can then be naturally seen as the transmission of items from the data base of aggregations of values from the data base. So we can have:

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

or summaries

$$Gossip(\alpha, \gamma, \langle \beta, negotiation, seller, Very\_good\rangle)$$

[4]there might be more than one commitment affecting different messages of a particular im.

2.2 Semantic Similarity

Similarity measures Interactions between agents might be sparse in a large P2P society, so we exploit an imperfect reasoning method to maximise the information extracted from any interaction. The idea is that concrete experiences about commitments over $\langle IM, r, m, \varphi, c_m \rangle$ can be used to update the expectation of behaviour over semantically close commitments. The method is based on similarity measures along three dimensions.

- Similarity on vocabulary. 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. How commitments, *Commit*(·) about objects in a particular semantic region, and their execution *Observe*(·), *affect* our decision making process about accepting new commitments on nearby semantic regions is crucial to scale the model. The measure we use [2] bases the *semantic similarity* between two concepts on the path length induced by ':' (more distance in the ':' tree means less semantic similarity), and the *depth* of the subsumed concept (common ancestor) in the shortest path between the two concepts (the deeper in the hierarchy, the closer the meaning of the concepts). For agent α semantic similarity could then be defined as:

$$\mathrm{Sim}_\tau(\varphi, \varphi') = \begin{cases} 1 & \text{if } \varphi = \varphi' \\ e^{-\kappa_1 l} \cdot \frac{e^{\kappa_2 h} - e^{-\kappa_2 h}}{e^{\kappa_2 h} + e^{-\kappa_2 h}} & \text{otherwise} \end{cases} \tag{2}$$

  where l is the length (i.e. number of hops) of the shortest path between the concepts[5], h is the depth of the deepest concept subsuming both concepts, and κ_1 and κ_2 are parameters balancing the contribution of shortest path length and depth respectively.[6]

- Similarity on roles. Roles are described in OpenKnowledge as either (i) a number of keywords, or (ii) as a set of terms in the local ontology of the designer of the *im*. We will use the matching technology developed elsewhere in the project [2] or the method described above in the case of terms.

  – Keywords:

$$\mathrm{Sim}_\mathrm{r}(r, r') = \mathit{matching}(\mathit{Keywords}(r), \mathit{Keywords}(r')) \tag{3}$$

  – Terms:

$$\begin{aligned} \mathrm{Sim}_\mathrm{r}(r, r') = \frac{1}{2} \cdot (&\sum_{\varphi \in \mathit{terms}(r)} \frac{\mathit{max}\{\mathit{Sim}_\tau(\varphi', \varphi) \mid \varphi' \in \mathit{terms}(r')\}}{\mid \mathit{terms}(r) \mid} + \\ &\sum_{\varphi \in \mathit{terms}(r')} \frac{\mathit{max}\{\mathit{Sim}_\tau(\varphi', \varphi) \mid \varphi' \in \mathit{terms}(r)\}}{\mid \mathit{terms}(r') \mid}) \end{aligned} \tag{4}$$

[5]these hops have to take into account the type of the link, a link representing a *part-of* implies a higher semantic distance when traversing it than a link labelled with *is-a*.

[6][2] argues that $\kappa_1 \simeq 0.2$ and $\kappa_2 \simeq 0.6$ represent a good model of human intuitions about similarity.

- Similarity on interaction models. As in the case of roles, interaction models are described with a list of keywords or with a list of terms.
  - Keywords:
$$\mathrm{Sim}_{\mathrm{im}}(im, im') = matching(Keywords(im), Keywords(im')) \tag{5}$$
  - Terms:
$$\begin{aligned}\mathrm{Sim}_{\mathrm{im}}(im, im') = \frac{1}{2}\cdot(&\sum_{\varphi \in terms(im)} \frac{max\{Sim_{\tau}(\varphi', \varphi) \mid \varphi' \in terms(im')\}}{\mid terms(im) \mid} + \\ &\sum_{\varphi \in terms(im')} \frac{max\{Sim_{\tau}(\varphi', \varphi) \mid \varphi' \in terms(im)\}}{\mid terms(im') \mid})\end{aligned} \tag{6}$$

Message semantics The semantics of messages is determined in LCC as the relationship between the content of a message and the constraints that the agent has to satisfy and that guard the sending of the message. By looking into the LCC code of an atomic interaction model *im* we can obtain the set of constraints that have to be satisfied in order for m to be sent. However, in cases where the peer has indicated through mapping that it will be performing not the precise constraints detailed in the *im* but a mapping of these constraints to its own abilities, the semantics of the message is then determined by these mappings. In order to simplify the amount of search that is necessary to determine what abilities a peer claimed to have, we therefore extend c_m to include all constraints that the peer is promising to satisfy, whether these are perfect mappings or not. This therefore removes the necessity to look inside both φ and the *im* to calculate those c_m a peer agreed to fulfil.

Matching Experience will tell us in which different contexts (i.e. *im*s) the peer has been *capable* of uttering particular message contents given their concrete relation (through an *im*) with certain constraints. It is therefore the history of interactions that tells us what 'semantics' the peer under scrutiny can understand. We can define the constraints we know agent β can deal with as:

$$H_{\beta} = \{c_m \mid \langle \beta, IM, \{\ldots Commit(\beta, \_, \langle im, r, m, \varphi, c_m \rangle, \varphi', \_, d) \ldots\}, \_\rangle \in M_{\beta}\}, value(d) \geq \zeta\} \tag{7}$$

Thus, given a $Commit(\beta, \alpha, \langle im, r, m, \varphi, c_m \rangle)$ α needs to assess whether β is capable of doing φ, that is to estimate $P(Can(\beta, \langle im, r, m, \varphi, c_m \rangle))$. The way we propose is to look in the history of satisfactory (over a threhsold ζ) experiences for all constraints that were ever used/satisfied by the peer. Then, compute the current needs, i.e. the current constraints to be satisfied, and assess whether we believe the peer can satisfy them by looking for the past constraint maximally similar to each of the current ones.

$$P(Can(\beta, \langle im, r, m, \varphi, c_m \rangle)) = \prod_{c \in c_m} max\{Sim_{\tau}(c', c) \mid c' \in H_{\beta}\} \tag{8}$$

Overall similarity Thus, given all these previous considerations it is easy to define the similarity between the content of commitments as an aggregation (we model it as a weighted combination depending on parameters γ_{im}, γ_r, γ_φ, and γ_m and γ_{c_m}) of several similarity measures:

$$\begin{aligned} &Sim(\langle im, r, m, \varphi, c_m\rangle, \langle im', r', m', \varphi', c'_m\rangle) = \\ &\quad Sim_{im}(im, im')^{\gamma_{im}} \cdot Sim_r(r, r')^{\gamma_r} \cdot Sim_\tau(\varphi, \varphi')^{\gamma_\varphi} \cdot Sim_\tau(m, m')^{\gamma_m} \cdot Sim_c(c_m, c'_m)^{\gamma_{c_m}} \end{aligned} \tag{9}$$

where the last factor, that is, similarity between sets of constraints, is defined as:

$$Sim_c(c_m, c'_m) = \left(\prod_{c \in c_m} \max_{c' inc'_m} Sim_\tau(c, c')\right) \cdot \left(\prod_{c \in c'_m} \max_{c' inc_m} Sim_\tau(c, c')\right) \tag{10}$$

NOTE: In the text of the algorithm 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 $Sim(\varphi, \varphi')$.

2.3 Decay

The integrity of percepts decreases in time. α may have background knowledge concerning the expected integrity of a percept 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*. If the background knowledge is incomplete then one possibility is for α to assume that the decay limit distribution has maximum entropy whilst being consistent with the data. Given a distribution, $\mathbb{P}(X_i)$, and a decay limit distribution $\mathbb{D}(X_i)$, $\mathbb{P}(X_i)$ decays by:

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

where Δ_i is the *decay function* for the X_i satisfying the property that $\lim_{t\to\infty} \mathbb{P}^t(X_i) = \mathbb{D}(X_i)$. For example, Δ_i could be linear: $\mathbb{P}^{t+1}(X_i) = (1 - \nu_i) \times \mathbb{D}(X_i) + \nu_i \times \mathbb{P}^t(X_i)$, where $\nu_i < 1$ is the decay rate for the i'th distribution. The decay function or the decay limit distribution could also be a function of time: Δ_i^t and $\mathbb{D}^t(X_i)$.

2.4 Update

Remember that:

$$\mathbb{P}(\varphi_i|\varphi) = P(Can(\varphi_i)) \cdot P(Does(\varphi_i)|Commit(\varphi))$$

and that we assessed $P(Can(\varphi_i))$ in the previous section.

In the absence of in-coming messages the integrity of $P(\varphi|\varphi)$ decays by Eqn. 11. The following procedure updates $P(\varphi|\varphi)$. Here we explain how to update $P(Does(\varphi_i)|Commit(\varphi))$, noted as $P(\varphi|\varphi)$ for simplicity.

Suppose that α has an experience $\mu = (\phi', \phi)$ of perceiving the execution ϕ' of a commitment ϕ and values it with d, then α must attach an epistemic belief $\mathbb{R}^t(\alpha, \beta, \mu)$ to

$P(\varphi|\varphi)$ — this probability reflects α's level of personal *caution* with what this experience means for the future. Thus the default definition:

$$\mathbb{R}^t(\alpha, \beta, (\phi', \phi)) = P(\phi'|\phi) + (1 - P(\phi'|\phi)) \cdot P(\phi'|\phi)$$

Denote the prior distribution $P^t(\varphi|\varphi)$ by $\vec{p}$, and let $\vec{p}_{(\mu)}$ be the distribution with minimum relative entropy[7] with respect to $\vec{p}$: $\vec{p}_{(\mu)} = \arg\min_{\vec{r}} \sum_j r_j \log \frac{r_j}{p_j}$ that satisfies the following constraints:

$$\begin{aligned} &\forall \varphi : sim(\varphi, \phi) \geq \omega \\ P(Does(\varphi_i)|Commit(\varphi)) &= P^t(\varphi'|\varphi) + 1/n_{ex} \cdot S \cdot (1 - P^t(\varphi'|\varphi)) \end{aligned} \tag{12}$$

where $S(\phi', \phi, \varphi) = (1- \mid \mathrm{Sim}(\phi', \phi) - \mathrm{Sim}(\varphi_i, \varphi) \mid) \cdot \mathrm{Sim}(\varphi, \phi)$, and $Sim(\theta, \psi)$ $= Sim(\langle IM_\theta, r_\theta, m_\theta, \theta, c_m\rangle, \langle IM_\psi, r_\psi, m_\psi, \psi, c_m\rangle)$.

Where n_{ex} represents the minimum consequent good experiences to have a high confidence on the willingness of the oponent. Finally, using Eqn. 11 we obtain the method for updating the distribution $\mathbb{P}^{t+1}(\varphi|\varphi)$ after the observation $\mu = (\phi', \phi)$:

$$\mathbb{P}^{t+1}(\varphi|\varphi) = \Delta_i(\mathbb{D}(X_i), \vec{p}_{(\mu)}) \tag{13}$$

2.5 Reputation

We propose two ways of modelling reputation. The first one based on sharing personal experiences among the peers to improve the trust computation and the second on a rating mechanism that can be used as an alternative to the trust measure altogether.

Gossip Basically gossip generates new constraints into the probability distributions. A peer 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 $\mu = (\phi', \phi)$ with its associated value d passed by a peer in the network, what is the value of $\mathbb{R}^t(\alpha, \beta, \mu)$? Once this is assessed then the method would be as before.

The assessment can be based on social network analysis as exploited in the REGRET system [5]. We don't explore this in detail here.

RepuRank Peers can rate each other's activities (that is, in our context, satisfaction of the commitments in an *im*) together with the certainty of their ratings. For example, a peer receiving a MP3 file transformed into MIDI may not be satisfied with the result

[7] Given a probability distribution $\vec{q}$, the *minimum relative entropy distribution* $\vec{p} = (p_1, \ldots, p_I)$ subject to a set of J linear constraints $\vec{g} = \{g_j(\vec{p}) = \vec{a_j} \cdot \vec{p} - c_j = 0\}, j = 1, \ldots, J$ (that must include the constraint $\sum_i p_i - 1 = 0$) is: $\vec{p} = \arg\min_{\vec{r}} \sum_j r_j \log \frac{r_j}{q_j}$. This may be calculated by introducing Lagrange multipliers $\vec{\lambda}$: $L(\vec{p}, \vec{\lambda}) = \sum_j p_j \log \frac{p_j}{q_j} + \vec{\lambda} \cdot \vec{g}$. Minimising L, $\{\frac{\partial L}{\partial \lambda_j} = g_j(\vec{p}) = 0\}, j = 1, \ldots, J$ is the set of given constraints $\vec{g}$, and a solution to $\frac{\partial L}{\partial p_i} = 0, i = 1, \ldots, I$ leads eventually to $\vec{p}$. Entropy-based inference is a form of Bayesian inference that is convenient when the data is sparse [3] and encapsulates common-sense reasoning [4].

but is not sure whether it is the fault of the peer offering the service or whether it is due to the low quality of the MP3 file. The ratings that peers can give is a pair of values $\langle x, y \rangle$, where the first value $-1 \leq x \leq 1$ is the indication of quality, ranging from very bad (-1), neutral (0) to very good (1) and the second value $0 \leq y \leq 1$ is an indication of how certain the peer was on the provided rating, ranging from completely unsure (0) to very sure (1). We assume peers are willing to give ratings, for example based on altruism, revenge or their own benefit. Ratings may also be given automatically in case that a peer is represented by a computer. For example, a peer α may give a negative ranking about a peer β if communication is expected at a certain moment and β does not respond.

These ratings influence the *authority* of an agent. The better its ratings the more authority an agent has. Also, the more authority an agent has the more influence its opinion has on the authority of others. In this way an algorithm similar to PageRank is obtained.

2.6 Trust equations

We define three different trust equation systems that can be implemented (we give later an algorithm implementing the second one), or chosen, depending on the particular personality of the peer.

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: $\mathbb{P}^t_I(\varphi'|\varphi)$. Here we measure the relative entropy between this ideal distribution, $\mathbb{P}^t_I(\varphi'|\varphi)$, and the distribution of expected enactments, $\mathbb{P}^t(\varphi'|\varphi)$. That is:

$$T(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle) = 1 - \sum_{\varphi'} \mathbb{P}^t_I(\varphi'|\varphi) \log \frac{\mathbb{P}^t_I(\varphi'|\varphi)}{\mathbb{P}^t(\varphi'|\varphi)} \tag{14}$$

where the "1" is an arbitrarily chosen constant being the maximum value that this measure may have. This equation measures one, single commitment φ.

Preferred enactments. The previous measure, 'Ideal enactments', requires that an ideal distribution, $\mathbb{P}^t_I(\varphi'|\varphi, e)$, has to be specified for each φ. Here we measure the extent to which the enactment φ' is preferable to the commitment φ. Given a predicate $\text{Prefer}(c_1, c_2, e)$ meaning that α prefers c_1 to c_2 in environment e. Then:

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

Satisfactory enactments. Here we measure the extent to which the enactment φ' given the commitment φ satisfies a need χ. Given a predicate $\text{Sat}(\varphi', \varphi, \chi)$ meaning that α gets its need χ satisfied by φ' in the context of commitment φ, we can define:

$$T(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle, \chi) = \sum_{\varphi'} \mathbb{P}^t(\text{Sat}(\varphi', \varphi, \chi))\mathbb{P}^t(\varphi' \mid \varphi)$$

this probability distribution can be assessed by taking into account the value d in the history of interactions.

Certainty in enactment. Here we measure the consistency in expected acceptable enactment of commitments, or "the lack of expected uncertainty in those possible enactments that are better than the commitment as specified". Let: $\Phi_+(\varphi, \kappa) = \{\varphi' \mid \mathbb{P}^t(\text{Prefer}(\varphi', \varphi)) > \kappa\}$ for some constant κ, and:

$$T(\alpha, \beta, \langle im, r, m, \varphi, c_m \rangle) = 1 + \frac{1}{B^*} \cdot \sum_{\varphi' \in \Phi_+(\varphi, \kappa)} \mathbb{P}^t_+(\varphi'|\varphi) \log \mathbb{P}^t_+(\varphi'|\varphi)$$

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

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

Aggregations. For an overall estimate of α's *trust* on β using any of the previous measures the procedure consists on projecting from M_β those experiences (including possibly gossips) that are relevant to the measure to be obtained. For instance in the previous trust measures we computed the $P(\varphi \mid \varphi)$ (in fact $P(Observe(\alpha, \beta, \varphi)|Commit(\beta, \alpha, \varphi))$ —remember the notation simplification) obtaining the experiences for a particular im, r, m and φ. If we want to compute other trust measures we simple focus on the appropriate subset of experiences. For instance $T(\alpha, \beta, \langle im, r \rangle)$ will be computed by looking into all experiences for im and r in the data base. Measures that take into account the importance of certain terms (or ims or rs) are also easy to define. Imagine a function that gives the importance of terms $f : Terms \to [0, 1]$ that is normalised. We can then define an aggregation as:

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

where $\mathbb{P}^t_\beta(\varphi)$ is a probability distribution over the space of commitments that the next commitment β will make to α is φ.

2.7 Trust algorithm

We give, as an example of the default trust algorithms provided by OpenKnowledge, the one that uses Minimum Relative Entropy and preferences, as illustrated in Algorithm 1. We assume equiprobable distributions for the decay limit distributions and linear decay. The other default algorithms are 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 memoizing can help in increasing the efficiency of the algorithm.

Calendar time can be simulated by adding an empty μ to M_α for each instant of time with no experiences.

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

3 Integration of Matching and Trust

As can be seen from the discussion in the previous section, there is a close relationship between matching and trust because evaluation of similarities between constraints, through matching, is crucial to the output of the trust process. In this section, we discuss the integration between matching and trust and how we can use both of these to identify a peer that is likely to be the best at performing a given role.

The matching process determines how well a peer's abilities map to the constraints on a role it wishes to play and results in a score and a set of mappings c_m. Since the mappings produced by the process define the way in which the peer will satisfy the constraints during the interaction, this matching score gives an accurate reflection of how well that peer believes it will be able to satisfy the constraints. However, it is important to modify the matching score reported by a peer with some kind of trust measure because:

- The peer may have a poorly organised local ontology, and its understanding of how closely matched its abilities and the constraints are may be flawed;
- The score returned by the matching process will accurately reflect the peer's understanding of how well it can satisfy a constraint; however, the peer may decide to cheat and report a different score during the subscription process. Since the matching score depends on a private local ontology and cannot be externally verified, it cannot be known whether the peer is being honest about its capabilities.
- As discussed previously, a peer's willingness to perform an action may not match its ability to do so: it may commit to perform an action that it is capable of performing well but nevertheless perform it poorly.

Section 2.1 describes how an assessment of both the capabilities and willingness of a peer with reference to a particular constraint (or ability) is crucial in determining the trust score. Nevertheless, computing the capability according to Equation 8 is not enough. The matching process is independently useful because:

- What is being measured of an *im* is different. The matching process measures the capability of a peer based on the similarity between the constraints and the abilities of a peer that may perform a role (say, $peer_1$), judged according to $peer_1$'s ontology. Computing the capability aspect of the trust process measures how similar a constraint is to a constraint that $peer_1$ has been known to (attempt to) satisfy previously, judged according to the ontology of a peer that may choose to interact with $peer_1$ (say, $peer_2$). $peer_2$'s judgement of similarity of constraints may not correspond to $peer_1$'s, since they are based on different ontologies, so although, if

Algorithm 1 function Trust(α, β, $\langle im, r, m, \varphi, c_m \rangle$)

```
**Require:** $O$ {the peer finite local ontology}
**Require:** $\kappa_1, \kappa_2 : Real$ [Default 1.0] {parameters of the similarity function}
**Require:** $\eta : [0,1]$ [Default 0.8] {min. sem. similarity in the computation}
**Require:** $\nu : [0,1]$ [Default 0.95] {decay parameter}
**Require:** $\zeta : [0,1]$ [Default 0.7] {minimum satisfaction}
**Require:** $\omega : [0,1]$ [Default 0.1] {minimum combined similarity}
**Require:** $n_{ex} : \mathbb{N}$ [Default 6] {number of experiences to be highly confident}
**Require:** $Prefer : Terms(O) \times \{\varphi\} \rightarrow [0,1]$ [Default $Prefer(x, y) = if\ x = y\ then\ 1\ else\ 0$] {a prob.
  dist. for preference of terms over $\varphi$ represented as a vector}
**Require:** $M_\alpha \subseteq M$ {$\alpha$'s log of experiences sorted by time}
**Ensure:** $Trust(\alpha, \beta, \langle IM, r, m, \varphi, c_m \rangle) \in [0,1]$
  $Focus \leftarrow \emptyset$
  for all $\varphi'$ in Terms(O) do
    if $Sim_\tau(\varphi', \varphi, \kappa_1, \kappa_2) \geq \eta$ then
      $Focus \leftarrow Focus \cup \{\varphi'\}$
    end if
  end for{we assume finiteness of Terms(O)}
  for all $\varphi'$ in Focus do
    $D(\varphi' \mid \varphi) \leftarrow 1/size(Focus)$
  end for
  $H_\beta = \emptyset$
  for all $\mu = \langle \beta, IM, PC, C \rangle$ in $M_\alpha$
      and $\langle Commit(\beta, \_, \langle im, r, m, \varphi', c' \rangle, t), \varphi'', 1, d \rangle \in PC$ with $d \geq \zeta$ do
    $H_\beta = c' \cup H_\beta$
  end for
  $Match \leftarrow 1$
  for all $\phi \in c_m$ do
    $MAX \leftarrow 0$
    for all $\phi' \in H_\beta$ do
      $MAX \leftarrow \max\{MAX, Sim(\phi, \phi')\}$
    end for
    $Match \leftarrow MAX \cdot Match$
  end for
  $t \leftarrow 0$; $P^t = D$
  for all $\mu = \langle \beta, im, PC, C \rangle$ in $M_\alpha$
      and $\langle Commit(\beta, \_, \langle im, r, m, \varphi_c, c_m \rangle, t), \varphi'', 1, d \rangle \in PC$ do
    if $Sim(\varphi_c, \varphi) \geq \omega$ then
      for all $\varphi'$ in Focus do
        $S \leftarrow (1 - \mid Sim(\varphi'', \varphi_c) - Sim(\varphi', \varphi) \mid) \cdot Sim(\varphi_c, \varphi)$
        $Q(\varphi' \mid \varphi) \leftarrow Match \cdot (P^t(\varphi' | \varphi) + 1/n_{ex} \cdot S \cdot (1 - P^t(\varphi' | \varphi)))$ {Constraints to satisfy}
      end for
      $P^{t+1} \leftarrow (1 - \nu)D + \nu \cdot MRE(P^t, Q)$ {MRE is the Min. relative entropy from $P^t$ satisfying $Q$}
    end if
    $t \leftarrow t + 1$
  end for
  $T \leftarrow 0$
  for all $\varphi'$ in Focus do
    $T \leftarrow T + Prefer(\varphi', \varphi) * P^{t-1}(\varphi' \mid \varphi)$
  end for
  return $T$
```

$peer_2$ judges constraint c_1 and constraint c_2 to be very similar, it would therefore consider $peer_1$'s prior performance on c_1 to be highly indicative of its performance on c_2, $peer_1$ may in fact map them to different abilities and the expected performances will not be correlated. In both situations, we are forced to make judgements based on incomplete information, but different kinds of incomplete information in each case. Therefore by allowing them both to be factors in our overall judgement of suitability of peers for roles, we maximise the chances of an accurate outcome.

- From a pragmatic point of view, using both the matching score and the trust score to choose a peer spreads the workload more evenly. Every peer must give its matching score when subscribing to a role (along with the list of constraints it is intending to satisfy, c_m, which is derived from the matching process). Therefore, this information comes free to any other peer potentially involved in the interaction. However, trust scores must be calculated by every peer for every other peer they may potentially interact with, and is thus an expensive process. By utilising this free matching information, peers can reduce the number of trust calculations they need to perform by only assessing peers that have a reasonably high matching score and ignoring peers that are unlikely to be able to perform the role well.

Although this illustrates that an assessment of both matching and trust scores is important in evaluating expected behaviour of peers, we cannot expect to use these scores to determine precisely how a peer will behave in a given situation. However, trust and matching scores are orthogonal, which makes it hard to define a precise semantics of the combination of matching and trust scores. Furthermore, this relation is context dependent and is based on the preference of the peer and its hidden decision processes.

We therefore suggest that the only way to determine how to combine these two scores in a way which will most often lead to us identifying the best peer for a role correctly or with reasonable reliability is through pragmatic evaluation of the behaviour of peers. The incomplete and unreliable nature of the tasks means we cannot give a formal underpinning of this combination but can merely say that experiment indicates that a certain method of integration is generally the most successful. We therefore leave the question as to exactly how this combination will be calculated to a stage when thorough empirical evaluation of peer behaviour is possible. In the meantime, we outline various approaches which we intend to test as soon as possible.

In any approach we take, we will want to use the matching scores provided by peers as a filter, so as to avoid performing expensive trust calculations for more peers than necessary.

Combining matching and trust scores: Using this approach, illustrated in Algorithm 2, we first remove all peers with a matching score below a certain threshold and then, for the remaining peers, calculate their trust scores. The *best_peer* is the one for which the combination of the trust and matching scores is highest, calculated according to the equation $t_i.\nu + m_i.(1-\nu)$, where t_i is the trust score for $peer_i$, m_i is the 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_i and m_i. The optimal value of ν may either be decided by the peer based on the context or may be determined empirically. Moreover, the suitability of any form of this equation will need to be determined empirically; it may be that a linear combination of the two scores is not the best approach.

Determining intervals: Using this approach, illustrated in Algorithm 3, peers are sorted into bands of width ψ ($\psi \in [0...1]$) according to their matching scores. Thus the first band would contain any peer with perfect matching; the second band would contain any peer with matching scores $(1 - \psi) \leq m_i < 1$, 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_peer* is the one from the highest matching band that has the best trust score, assuming that this trust score exceeds some basic threshold ζ. If the highest trust score for a peer from the highest matching band does not exceed this threshold, these peers are rejected and the next matching band is inspected, until a suitable peer is found.

We intend to implement these algorithms, and potentially further algorithms, and evaluate their relative merits on their performance. It is possible that testing may indicate that neither approach is optimal and point us towards an improved method of integration of matching and trust.

Algorithm 2 Find best peer for a role r in interaction model IM by combining matching and trust scores, as calculated by peer α

```
Require: $P$ {the set of potential peers}
Require: $\forall p_i \in P.m_i \in [0,1]$ {where $m_i$ is the matching score declared by $p_i$ on subscribing to the role.}
Require: $\xi : [0,1]$ {the matching threshold}
Require: $\nu : [0,1]$ {trust/matching weighting variable}
  $poss\_peers \leftarrow \{p_i \mid p_i \in P, p_i.m_i \geq \xi\}$
  $highest\_score \leftarrow 0$
  $best\_peer \leftarrow nil$
  for all $p_i \in poss\_peers$ do
    $p_i.t_i \leftarrow trust(\alpha, p_i, \langle IM, r\rangle)$
    $p_i.score \leftarrow p_i.t_i.\nu \cdot p_i.m_i.(1-\nu)$
    if $p_i.score > highest\_score$ then
      $highest\_score \leftarrow p_i.score$
      $best\_peer \leftarrow p_i$
    end if
  end for
  return best_peer
```

4 Lifecycle

This section illustrates where assessments of good enough performance are necessary through tracing the lifecycle of an interaction from the point of view of a particular peer, named $peer_1$.

Algorithm 3 Find best peer for a role r in interaction model IM through threshold lowering, as calculated by peer α

```
Require: $P$ {a set of peers}
Require: $\forall p_i \in P.m_i \in [0,1]$ {where $m_i$ is the matching score declared by $p_i$ on subscribing to the role.}
Require: $\psi : [0,1]$ {threshold lowering interval}
Require: $\zeta : [0,1]$ {trust threshold}
  $poss\_peers \leftarrow \emptyset$
  $peers\_to\_inspect \leftarrow P$
  $threshold \leftarrow 1$
  $highest\_trust \leftarrow 0$
  $best\_peer \leftarrow \emptyset$
  while $best\_peer = \emptyset$ do
    $peers\_to\_inspect \leftarrow P - poss\_peers$
    while $poss\_peers = \emptyset$ do
      for all $p_i \in peers\_to\_inspect$ do
        if $m_i \geq threshold$ then
          $poss\_peers \leftarrow poss\_peers \cup p_i$
        end if
      end for
      $threshold \leftarrow threshold - \psi$
      $poss\_peers \leftarrow \emptyset$
    end while
    for all $p_i \in poss\_peers$ do
      $trust(\alpha, p_i, IM, r) = t_i$
      if $t_i \geq \zeta$ then
        if $t_i > highest\_trust$ then
          $highest\_trust \leftarrow t_i$
          $best\_peer \leftarrow p_i$
        end if
      end if
    end for
  end while
  return best_peer
```

- $peer_1$'s user decides he wishes to play the role of a buyer in a buying-selling interaction. He therefore inputs the following query into the discovery service (DS):

  Buying selling interaction

  The user, if he wishes, may also annotate this with keywords that he considers appropriate. For example, if he wants to buy a computer, he may add the keyword *computer*. This keyword would usually be annotated with a reference to the ontology it is from (which could be a URI or a path from the ontology) in order to give it context; however, this is not compulsory. This keyword is not attached to any specific variable - this would be impossible as the IM has not yet been chosen - but is intended to give non-specific context to the interaction.

- The DS returns a list of potentially suitable IMs, ranked according to various criteria such as:

  - How well the query of the user matches the keyword description of the IM. This analysis also includes consideration of any annotation keywords added by the user, so, in this case, an IM described as $computer-buying-selling$ would match better than one labelled simply $buying - selling$.
  - Some evaluation of how popular the IM is. The precise mechanics of this have not yet been determined, but the RepuRank process discussed in Section 2.5 could be used to gather reputation information about IMs. Additionally, information services could be set up that would monitor how frequently IMs were used and possibly to record any user feedback on them.
  - How many peers are already subscribed to the IM. If an IM has no peers subscribed to other roles, it cannot be immediately executed once $peer_1$ has joined. Also, the more peers that are subscribed to the roles in an IM, the more choice $peer_1$ has about whom to interact with. However, if there are already many peers subscribed to the role that $peer_1$ wishes to play, then this could be seen as a disadvantage, as $peer_1$ is less likely to be chosen by other peers to play its desired role.

  This process returns a ranked list of IMs to $peer_1$, with the highest ranked IMs most likely to be suitable for $peer_1$'s needs. Along with every IM will be details of peers subscribed to each role in the IM, including their reported matching score and the details of exactly what constraints these peers intend to satisfy, where this differs from the constraints on the role they are subscribed to playing (this constraint mapping process is described in more detail in the next step, where we discuss how $peer_1$ determines its constraints).

  This judgement of suitability is very general, depending on keyword descriptions and the experiences of other peers, and in order to tell more precisely how suitable the IM is, $peer_1$ must compare the abilities and consequences required in the role it wishes to play (the constraints on that role) with its own abilities and goals. This is done through the matching process. Since matching is fairly expensive, $peer_1$ can use the automatic ranking so that it performs matching only on highly ranked IMs.

$a(buyer, B) ::$

$$\begin{array}{rcll}
ask(C, Ma, Mo, Y) & \Rightarrow & a(seller, S) & \leftarrow \; need(C : car, Ma : make, Mo : model, Y : year) \\
price(C, Ma, Mo, Y, P) & \Leftarrow & a(seller, S) & then \\
buy(C, Ma, Mo, Y, P) & \Rightarrow & a(seller, S) & \leftarrow afford(C, P : Price) \\
owns(C, Ma, Mo, Y) \leftarrow sold(C, P) & \Leftarrow & a(seller, S) &
\end{array}$$

$a(seller, S) ::$

$$\begin{array}{rcll}
ask(C, Ma, Mo, Y) & \Leftarrow & a(buyer, B) & then \\
price(C, Ma, Mo, Y, P) & \Rightarrow & a(buyer, B) & \leftarrow instock(C, Ma, Mo, Y, P) \\
buy(C, Ma, Mo, Y, P) & \Leftarrow & a(buyer, B) & then \\
sold(C, P) & \Rightarrow & a(buyer, B) &
\end{array}$$

$C = Colour; Ma = Make; Mo = Model; Y = Year$

Figure 1: Buying-selling IM

$peer_1$ can then re-rank the IMs according to its matching scores and choose the one that has the highest score.

The top ranked IM is the one shown in Figure 1. In this figure, we indicate the types of the arguments through using the type name as the argument name: for example, $need(Item)$ indicates that the object that instantiates the variable $Item$ should be of type *item*.

- $peer_1$ performs automated matching between its OKC (which describes the actions it can perform when it is playing the part of buyer) and the constraints that are on the role of buyer in the top ranked IMs.

Example 1 (Perfect Matching) *$peer_1$ has a buying OKC that is able to interpret the following constraints:*
require(Model,Make,Year,Car)
afford(Item,Price)

$peer_1$ uses the matcher to determine how similar these abilities are to the constraints. The matcher can trivially determine that afford(Item,Price) *is a perfect match for* afford(Car,Price) *because Item can be instantiated to the car function, and simple matching can indicate that* require(Model,Make,Year,Car) *is semantically and syntactically identical to* need(Car,Make,Model,Year) *because* require *and* need *can be determined to be semantically identical and the ordering of arguments is not semantically significant (though in reality, the context of these terms may be needed to determine that they are identical rather than merely similar).*

$peer_1$ can therefore determine that its OKC is a perfect match for the buyer role described in Figure 1.

It will report a matching score of 1 to the subscription process. The peer must also develop the set c_m and the messages $\varphi \in \Phi$ that it plans to commit to. We therefore have:

$c_{m_1} = \{require(Mo, Ma, Y, C)\}$
$c_{m_2} = \{afford(Item, P)\}$

$\varphi_1 = ask(C : car, Ma : make, Mo : model, Y : year)$
$\varphi_2 = buy(C : car, Ma : make, Mo : model, Y : year, P : price)$

Any reference to objects in $peer_1$'s ontology can be annotated with a reference to that ontology: either a URI where the entire or part of the ontology can be found or a path from within the ontology. Such ontological annotation is not essential; however, a peer that does not properly explain its terms in such a manner is likely to be less preferred than one that does, since it is less easy for other peers to work out the consequences of interacting with such a peer. As we have seen before, other peers have no way of verifying that $peer_1$ will actually use the ability $require(Mo, Ma, Y, C)$ to satisfy the constraint $need(C, Ma, Mo, Y)$; it may be that he has no such ability and is lying. This is one of the reasons why it is important to factor in a trust score.

Example 2 (Approximate Matching) *$peer_1$ has a buying OKC that is able to interpret the following constraints:*
require(Auto,Model,Make)
afford(Item,Quantity,Price)

The matcher cannot determine a perfect match between these abilities and the constraints on the buying role. For the first constraint, it can determine that require *matches* need *and that* Auto *matches* Car *but the constraint has an extra argument* Year *that has no corresponding argument in the ability. Conversely, $peer_1$'s ability* afford(Item,Quantity,Price) *has an extra argument* Quantity *that is not mirrored in the constraint.*

However, it can find an approximate match. $peer_1$'s require *can match to the constraint* need *if the* Year *argument is dropped, and likewise $peer_1$ can ignore its* Quantity *argument to find a perfect match for* afford. *The matcher will provide a quantitative score for how similar these approximate matches are (the process by which this is done is not discussed here) and the average score of all these matches determines $peer_1$'s matching score for this role. If this matching score is low, $peer_1$ will probably not want to participate and will attempt to find a better IM. If the score is high enough that $peer_1$ still wishes to participate, he may do so but must make this matching score publicly available.*

The matching process will produce a score $0 \leq m_{peer_1}(c) < 1$ for the quality of the match on each constraint c: perhaps 0.8 for the first constraint and 0.85 for the second. How to combine these two scores is a complicated issue that depends on the whether these constraints lie on a choice branch of the interaction model's state-space or not. That is, failing to fulfil this constraint will definitely break the interaction or not, and so on. This issue should be dealt with by the matcher, which we do not discuss in this paper. To simplify our example, we assume we can take the average of these scores, so $peer_1$ (assuming it was honest) would report a score of $m_{peer_1} = 0.825$. It must also, as in the exact matching example, report c_m and the messages $\varphi \in \Phi$ that it plans to commit to. In this case,

$c_{m_1} = \{require(Auto, Mo, Ma)\}$
$c_{m_2} = \{afford(Item, Quantity, Price)\}$

$\varphi_1 = ask(C : auto, Ma : make, Mo : model, Y : \perp)$
$\varphi_2 = buy(C : item, P : price)$

The reported mappings can also be used by other peers both to verify that they agree with the matching score and so that they can see where these areas of inexactness lie.

- *peer*$_1$ subscribes as a buyer to IM1. In order to subscribe, it must provide its matching score and details about where it failed to match as discussed above.

  As mentioned earlier in this document, commitments must be to messages rather than to constraints because message passing can be directly observed whereas constraint satisfaction cannot. However, the semantic content of messages is given by the constraints on them and so in practice what it means to commit to a particular message is to commit to giving the message a particular semantics through instantiating the variables passed in that message in a particular manner; that is, according to the constraints or, if the peer is not able to satisfy the constraints exactly, according to the mappings provided by the peer.

  Therefore, although commitment is really to messages, one can view these mappings as constituting the commitments the peer is making. It should not be blamed for failing to exactly satisfy the constraints on an IM if it has committed to satisfying an altered version of the constraint; it can only be expected to satisfy what it has committed to. As discussed earlier, peers can never directly observe how other peers are satisfying constraints. However, much information about this can be deduced from type checking the variables in the message against the mapped constraints and peers are therefore able to make educated guesses as to whether the peers adhered to their commitments to these mapped constraints.

  Note that *peer*$_1$ is not proposing a change to the IM as described in Figure 1; peers cannot change IMs on-the-fly during interactions. Instead, what the peer is saying is that it will satisfy the constraint - thus instantiating the variables - not by satisfying the expected constraint $need(car(Make, Model, Year, Colour))$ but by satisfying the similar constraint $require(car(Make, Model, Colour))$. Since the $Year$ variable does not appear in this new constraint, this will not be instantiated and an uninstantiated variable will be passed in the message (if the new constraint had an extra argument, it would not be possible to discuss this in the IM and it would therefore be ignored.) Other peers can judge whether a commitment to satisfy this new constraint will suffice; commitments to satisfy altered constraints may often lead to results and instantiations that are not desired by other peers.

- Imagine that every role in the IM now has peers subscribed to it; that is, there were already peer(s) subscribed to all the other roles in the IM: in this case, to the selling role, whereas the buying role was previously unsubscribed. Once a peer ($peer_1$) has subscribed to the buying role, the interaction is ready to commence.

- Let us imagine that the seller role has two peers subscribed to it: $peer_2$ and $peer_3$. These peers will have already provided their matching score and details of their precise commitments to the subscription service. This information is sent to the other peers potentially involved in the interaction.

- Where necessary (see Algorithms 2 and 3), trust scores must be calculated.

  There are many commitments involved in committing to a role: a commitment to each message to be passed, each of which can be reduced to a commitment to satisfy each of the constraints on that message in the manner specified in the matching results (or as specified in the IM if no matching information is given.) Users may, if they desire, weight particularly commitments more heavily than others because they feel that these commitments are particularly pertinent, though we would not expect such weighting to be the norm as it requires users to get involved at the level of message passing.

  To compute the trust measure on, for example, a message $sold(car)$ (in this example we use $sold(car)$ instead of $sold(C : car, Price : price)$ for simplification), the following formula should be calculated:

$$T(b, s, \langle imBS, seller, sold(car), sold(mercedes), \{instock(C, Ma, Mo, Y, P)\}\rangle) = \\ 1 - \sum_{\varphi'} \mathbb{P}_I^t(\varphi' | sold(mercedes)) \log \frac{\mathbb{P}_I^t(\varphi' | sold(mercedes))}{\mathbb{P}^t(\varphi' | sold(mercedes))} \tag{15}$$

  In the above equation, b is the buying peer interested in computing the trust score for the peer s playing the role of a *seller* in the buying selling interaction model *imBS* of Figure 1. The trust score is to be computed on the specific action of selling a car. Peer b is interested to know how much it can trust peer s to send it a *mercedes*. Note that while $sold(car)$ refers to the original message of the interaction model, the seller s could have committed to $sold(mercedes)$.

  To compute equation 15, an ideal distribution of enactments, $\mathbb{P}_I^t(\varphi' | sold(mercedes))$, needs to be defined. We propose a default distribution, which may be altered by the user to fit their particular requirements. In the default distribution, we assume that the only acceptable value is $sold(mercedes)$:

$$\mathbb{P}_I^t(\varphi' | sold(mercedes)) = \begin{cases} 1, & \varphi' = sold(mercedes) \\ 0, & otherwise \end{cases} \tag{16}$$

  Along with the ideal distribution of enactments, the realistic probability distribution, $\mathbb{P}^t(\varphi' | sold(mercedes))$, should also be obtained in order to compute the trust score of Equation 15. Note that this realistic distribution should be computed for all values φ' of the set $Focus$, the set of all expected terms. The set $Focus$ is constructed based on the ontology of the peer and the message expected in LCC. It basically represents what peer b thinks are all the possible messages that may be received. For example, if the message expected is $sold(car)$, the buying peer b may define $Focus$ as the set $\{sold(mercedes), sold(ford), sold(honda), sold(bmw), ...\}$.

In what follows, we show how the realistic probability distribution may be computed for one possible value of φ': $sold(ford)$. $\mathbb{P}^t(sold(ford)|sold(mercedes))$ is then computed accordingly:

$$\mathbb{P}^t(sold(ford)|sold(mercedes)) = \\ P(can(sold(ford))).P(does(sold(ford))|commit(sold(mercedes))) \tag{17}$$

$P(can(sold(ford)))$ represents the capability of performing $sold(ford)$. The database of previous experiences is consulted to see if the peer is capable of performing $sold(ford)$. The peer is said to be capable of performing $sold(ford)$ if it has performed this action at least once in the past. However, since capabilities are related to constraints rather than messages, the similarity is then computed on the constraints rather than the messages. This is done by making use of c_m, which is the constraint the peer is committing to (i.e. in our example, it is {instock(C,Ma,Mo,Y,P)}). Therefore, $P(can(sold(ford)))$ is computed as follows:

$$P(can(sold(ford))) = \prod_{\phi \in \{instock(C,Ma,Mo,Y,P)\}} max\{sim(\phi', \phi)|\phi' \in H_s\} \\ max\{sim(\phi', \{instock(C, Ma, Mo, Y, P)\})|\phi' \in H_s\} \tag{18}$$

where:

$$H_s = \cup\{c'_m| \\ \langle s, \_, \{..., \langle commit(s, \_, \langle im, r, m, \varphi_i, c'_m\rangle, \_), \varphi, g, d\rangle, ...\}, \_\rangle \in M_b \wedge value(d) \geq \zeta\} \tag{19}$$

In other words, the set H_s is the set of constraints that have been successful in the past, where success is measured through the threshold ζ.

After obtaining the probability of the peer being capable of performing $sold(ford)$, we now need to assess how much the peer's willingness in performing $sold(ford)$ is trusted given that it has committed to $sold(mercedes)$. The list of previous experiences μ_i of the buyer's database M_b is consulted. However, this time we need to study the results of all previous commitments similar to our current commitment $sold(mercedes)$ in order to obtain the probability of having $sold(ford)$ currently occur. Note that the database M_b might include the experience of other peers, which have been obtained through *gossip*, weighted according to how much reliable does the peer think this piece of gossip is.

Let us consider the following previous experience with a commitment relatively similar to $sold(mercedes)$:

$$\mu_x = \langle s, imX, \{\langle commit(s, \_, \langle imX, salesman, deliver(car), deliver(mercedes), \\ require(Make)\rangle, 15), deliver(bmw), 0, excellent\rangle\}, \_\rangle$$

Using the knowledge from experience μ_x, $P(sold(ford)|sold(mercedes))$ is then updated as follows (again, we use $P(sold(ford)|sold(mercedes))$ instead of $P(does(sold(ford))|commit(sold(mercedes)))$ for simplification):

$$P(sold(ford)|sold(mercedes)) = S \cdot \mathbb{R}^{15}(b, s, \mu_x) + (1 - S) \cdot \mathbb{P}^{15}(sold(ford)|sold(mercedes))) \tag{20}$$

where S represents the similarity between experiences, and is defined as:

$$\begin{aligned} S = (1 - |&Sim(deliver(bmw), deliver(mercedes)) - Sim(sold(ford), sold(mercedes))|) \cdot \\ &Sim(deliver(mercedes), sold(mercedes)) \end{aligned} \tag{21}$$

and $\mathbb{R}$ represents the reliability of the previous experience, which is defined as:

$$\begin{aligned} \mathbb{R}^{15} = &P(deliver(bmw)|deliver(mercedes)) + \\ &(1 - P(deliver(bmw)|deliver(mercedes))) \cdot P(deliver(bmw)|deliver(mercedes)) \end{aligned} \tag{22}$$

Assuming that the similarity between $deliver(bmw)$ and $deliver(mercedes)$ is 0.8, the similarity between $sold(ford)$ and $sold(mercedes)$ is 0.8, and the similarity between $deliver(mercedes)$ and $sold(mercedes)$ is 0.6. This implies that $S = (1 - |0.8 - 0.8|) * 0.6 = 0.6$. Also, assume that the probability of delivering a *bmw*, given that the peer committed to deliver *mercedes*, based on the new experince μ_x is 0.7. Therefore, $\mathbb{R}^{15} = 0.7 + (1 - 0.7) \cdot 0.7 = 0.91$. Finally, assume that the propabibility of $\mathbb{P}^{15}(sold(ford)|sold(mercedes))$ has been computed earlier and found to be 0.4[8]. As a result, Equation 20 will be evaluated as follows: $P(sold(ford)|sold(mercedes)) = 0.6 \cdot 0.91 + (1 - 0.6) \cdot 0.4 = 0.706$. Therefore, taking into consideration the experience μ_x, the probability that the seller will sell a *bmw*, given that it has committed to *mercedes*, is now raised from 0.4 to 0.706.

Finally, the time decay effect on the knowledge of the previous experience is then taken into consideration by applying the following:

$$\mathbb{P}^{t+1}(sold(ford)|sold(mercedes)) = (1 - 0.95)\frac{1}{size(Focus)} + 0.95.MRE(\mathbb{P}^{t-1}, \mathbb{P}) \tag{23}$$

where 0.95 is chosen to be the default time decay rate, the specification of MRE is left for future work.

Note that Equation 20 should be computed for all appropriate experiences of the database M_b. But since Equation 20 requires the computation of Equations 21 and 22, and its result is always updated by Equation 23, then Equations 20, 21, 22, and 23 should all be repeated for every appropriate experience in the database M_b. Also note that these computations require the use of trust and probability scores at previous timesteps. This implies that the experiences in database M_b should be sorted by time. For each time step (starting from timestep 0 and ending at the

[8]Note that the probability distribution is initially set to be $1/size(Focus)$. Then, at every time-step, it gets modified with new experiences. In this example, we assume the probability at timestep 15 has already been computed.

current timestep), Equation 20 (along with equations 21, 22, and 23) is computed for all experiences of that timestep. This is followed by the computation of Equation 17 (which require the computation of Equation 18 as well) for all possible outcomes of φ' of the set $Focus$. Finally, the final trust score of Equation 15 (which require the computation of Equation 16 as well) may be computed. One of the algorithms presented in Section 3 is then applied to take into consideration both the matching and trust scores in selecting the appropriate peer.

- Each peer must decide which other peers it wishes to be involved with. This decision is a combination of the matching score that each peer has provided and a trust score (as computed in the last step) for each peer. This combination is possibly done in the manner described in either Algorithm 2 or Algorithm 3.

  In the current kernel, peers are only allowed to make boolean yes/no decisions about other peers; they cannot rank them in order of preference. Peers can use Algorithms 2 or 3 to find the best peer, or to find the n best peers and agree only to work with them. Alternatively, these algorithms could be adapted so that they return any peer over a certain threshold. However, later adaptations of the kernel should allow peers to rank their preferences.

- Every time a peer receives a message, it is able to observe whether the peer that committed to send that message honoured its commitment or not. If the message is semantically correct according to the commitments made, this commitment is considered to be honoured and the function g (outcome) can be instantiated to $\top$. If the message is considered semantically incorrect or is not received, g is instantiated to $\bot$. However, as discussed previously, a failure to honour a commitment to passing a particular message actually means a failure to satisfy at least one of the constraints on that message in the way specified. It is, in the general case, not possible to pin this down to a failure of a particular constraint and therefore all constraints attached to that message must be blamed equally.

- This correct/incorrect judgement discussed in the previous point, which can be analysed automatically, is just one aspect of judging whether an interaction is successful or not. The other aspect of evaluating whether commitments have been honoured or not is the function d, which is instantiated to $\{bad, neutral, good, very\_good\}$ depending on user feedback. It is possible for a peer to correctly honour all its commitments and yet provide poor service; for example, selling a car that conforms to all the explicitly specified requirements and yet is somehow substandard. Therefore, feedback from the user is also very important in analysing how trustworthy a peer is. Feedback from the user would usually be given for a whole interaction rather than for a particular commitment to a message; users would not usually be involved in an interaction on such a low level. The overall judgement for an interaction can therefore be generalised to apply to each commitment. This may result in incorrect information for some commitments; however, it is not otherwise possible to factor in user feedback.

  For example, at the end of the interaction, the seller would add the following to its database of experiences:

$$\begin{aligned}\mu = \quad & \langle b, buy\_sell, \{\langle commit(s, b, \langle buy\_sell, seller, \\ & \qquad send(buy(C, Ma, Mo, Y, Price)), \\ & \qquad send(buy(C, Ma, Mo, Y, Price)), \\ & \qquad \{afford(Item, Quantity, Price)\}\rangle, 13), \\ & send(buy(C, Ma, Mo, Y, Price), 1, fair)\rangle\}, null\rangle\end{aligned}$$

This implies that in the interaction model *buy_sell*, the buyer b has committed to sending all appropriate messages. This includes the message $buy(C, Ma, Mo, Y, Price)$, the semantics of which are defined by the commitments to constraints that the buyer makes. Assuming that the buyer has provided the mappings discussed in Example 2 (approximate matching), this constraint will not be the one specified in the IM but the mapped constraint, which is therefore the one that is listed. If, at the end of the interaction, everything has gone well, the commitments' outcomes are therefore automatically given a mark of 1. However, if the seller wasn't entirely satisfied with the buyer's performance, then the user's subjective remark $fair$ is added. Note that *null* at the end of the tuple implies that no further commitments were produced by this interaction.

5 Conclusions and Further Work

This paper outlines our approach to the application of trust and matching judgements to interactions in order to make an assessment of whether an interaction is likely to be 'good enough'. However, much remains to be done:

- The question of exactly how commitments that arise as a consequence of interactions is dealt with must be explored in more detail. This issue is alluded to in this document but further exploration is necessary.
- The question of the combination of IMs to make compound IMs is also alluded to but not fully dealt with in this document. Judging whether subsequent IMs that must be committed to are likely to result in a 'good enough' outcome may be a difficult process and needs to be addressed in more detail.
- We need further research on how reputation will fit into trust measurements. We have already described how reputation may be shared through *gossip*. However, in order to determine how much credence is given to each piece of gossip, we need to assess issues such as social grouping to determine whether such information is likely to be unbiased. This problem has already been addressed in a thorough manner (cite Jordi's work and other relevant stuff) and we hope to incorporate much of this research into our assessment.
- The ideas within RepuRank need to be further developed and incorporated into the trust procedure.

- Extensive testing must be done to assess how successful these trust measures are in the OpenKnowledge context and to evaluate which method of combining matching and trust gives the best results. Achieving reliable results from the process requires a large number of disparate peers in different contexts to test on and therefore will be difficult to do until the system is well developed.

References

[1] Arcos, J.L., Esteva, M., Noriega, P., Rodrguez-Aguilar, J.A., Sierra, C.: Engineering open evironments with electronic institutions. Engineering applications of artificial intelligence. **18** (2005) 191 – 204

[2] Li, Y., Bandar, Z.A., McLean, D.: An approach for measuring semantic similarity between words using multiple information sources. IEEE Transactions on Knowledge and Data Engineering **15** (2003) 871 – 882

[3] Cheeseman, P., Stutz, J.: On The Relationship between Bayesian and Maximum Entropy Inference. In: Bayesian Inference and Maximum Entropy Methods in Science and Engineering. American Institute of Physics, Melville, NY, USA (2004) 445 – 461

[4] Paris, J.: Common sense and maximum entropy. Synthese **117** (1999) 75 – 93

[5] Sabater, J., Sierra, C.: Reputation and social network analysis in multi-agent systems. In: Proceedings of the First International Conference on Autonomous Agents and Multi-Agent systems. (2002) 475 – 482