

An Interoperable Context Sensitive Model of Trust

Indrakshi Ray Indrajit Ray Sudip Chakraborty

Department of Computer Science

Colorado State University

Fort Collins CO 80523-1873

{iray,indrajit,sudip}@cs.colostate.edu

Abstract

Although the notion of trust is widely used in secure information systems, very few works attempt to formally define it or reason about it. Moreover, in most works, trust is defined as a binary concept – either an entity is completely trusted or not at all. Absolute trust on an entity requires one to have complete knowledge about the entity. This is rarely the case in real-world applications. Not trusting an entity, on the other hand, prohibits all communications with the entity rendering it useless. In short, treating trust as a binary concept is not acceptable in practice. Consequently, a model is needed that incorporates the notion of different degrees of trust.

We propose a model that allows us to formalize trust relationships. The trust relationship between a truster and a trustee is associated with a context and depends on the experience, knowledge, and recommendation that the truster has with respect to the trustee in the given context. We show how our model can measure trust and compare two trust relationships in a given context. Sometimes enough information is not available about a given context to evaluate trust. Towards this end we show how the relationships between different contexts can be captured using a context graph. Formalizing the relationships between contexts allows us to extrapolate values from related contexts to approximate the trust of an entity even when all the information needed to calculate the trust is not available. Finally, we show how the semantic mismatch that arises because of different sources using different context graphs can be resolved and the trust of information obtained from these different sources compared.

1 Introduction

The concept of *trust* is widely used in secure information systems. For instance, entities participating in an e-commerce transaction must “trust” each other or rely on a “trusted” third party. However, there are no accepted formalisms or techniques for the specification of trust and for reasoning about trust. Secure systems have been built under the premise that concepts like “trustworthiness” or “trusted” are well understood, unfortunately without even agreeing on what “trust” means, how to measure it, how to compare two trust values and how to compose the same. This creates a number of inferential ambiguities in building secure systems, particularly those that are composed from several different components.

Consider, for example, the operational information base in a large corporation. Typically, this is generated by accumulating information from several sources. Some of these sources are under the direct administrative control of the corporation and thus are considered trustworthy. Other sources are “friendly” sources and information originating directly from them are also considered trustworthy. However, these “friendly” sources may have derived information from their own sources which the corporation does not have any first hand knowledge about; if such third-hand information is made available to the corporation, then the corporation has no real basis for determining the quality of that information. It will be rather naïve for the corporation to trust this information to the same extent that it trusts information from sources under its direct control. Similarly not trusting this information at all renders it useless. Existing binary models of trust (where trust has only two values, “no trust” and “complete trust” and which are the ones most widely used in computer systems) will, nonetheless, categorize the trust value to one of these two levels.

This motivated us to propose a new model of trust. In this model the trust relationship between the truster and the trustee can be of different degrees. Moreover, in the model, a trust relationship between a truster and a trustee is never absolute. The truster trusts the trustee with respect to a certain *context*. Trust between a truster and a trustee in a specific context is defined in the model as a 3-element vector. The elements correspond to factors, namely, *experience*, *knowledge* and *recommendation*, on which trust relationship depends. The value of each of these elements is represented as a triple represented as (b, d, u) where b , d , and u denote belief, disbelief, and uncertainty. The (b, d, u) notation is adopted from Jøsang’s opinion model [10, 11, 12]. We give some ideas about how to determine the values of belief, disbelief, and uncertainty component of each of these elements. In addition, we provide algorithms that show

how trust relationships expressed as vectors can be compared. We also investigate the dynamic nature of trust – how trust is not static but changes over time. Finally, we observe that trust depends on trust itself – that is a trust relationship established at some point of time in the past will influence the computation of trust at the current time.

The model that we described above is not very useful in computing trust relationships when the truster does not have any experience, knowledge, or recommendation about a trustee in a given context. An example will help to illustrate this. Suppose a user A (the truster) does not have any experience, knowledge or recommendation about the software developer B (the trustee) with respect to developing anti-virus software (the context). The model will not be able to evaluate the trust relationship in this case. Assuming that expertise to develop an anti-spam software (a related context) is similar to the expertise needed to develop anti-virus software, it seems natural that the trustee A will be able to determine how much to trust B for the different (but related) context. We extend the model to accommodate this situation.

This requires formalizing the notion of contexts. A context is described by a set of keywords. Multiple contexts may be related using generalization/specialization relationships or composition relationships. We show how these different relationships can be captured using a new data structure which we term as the *context graph*. Information obtained from different sources cannot, in general, be combined if they use different context graphs. However, we identify the relationships that can exist between different context graphs, and propose algorithms that enable us to combine different context graphs. Such formalization enables us to compare information obtained from different sources not all of which use the same terminology to describe a context. Moreover, when sufficient information is not available to calculate the trust vector in a given context, we show how to extrapolate the trust vector from a related context. This extrapolated vector can then be used to make some important trust related decisions.

The rest of the paper is organized as follows: Section 2 briefly describes some of the more important works in the area of trust models. Section 3 gives our definition of trust and provides an overview of our model. Section 4 defines the parameters on which trust depends and proposes techniques for assessing them. It also describes the concepts of normalized trust and the value of trust and discusses trust dynamics – the dependence of trust on time. This section also defines the dominance relation between two trust relationships that allows us to identify how to compare two trust relationships. Section 5 formalizes the notion of trust context

that will enable one to reason about trust relationships in different contexts. Section 6 discusses how values related to a trust relationship can be extrapolated when some information needed to compute the values is missing. Finally, section 7 concludes with a discussion of our future work.

2 Related Work

A number of logic-based formalisms of trust have been proposed by researchers. Almost all of these view trust as a binary relation. Forms of first order logic [1, 4, 8] and modal logic or its modification [17] have been variously used to model trust in these cases. Simple relational formulas like A trusts B are used to model trust between two entities. Each formalism extends this primitive construct to include features such as temporal constraints and predicate arguments. Given these primitives and the traditional conjunction, disjunction, negation and implication operators, these logical frameworks express trust rules in some language and reason about these properties. Abdul-Rahman and Hailes [1] propose a trust model, based on “reputation” that allows artificial agents to reason about trustworthiness and allows real people to automate that process. Jones and Firozabadi [9] model trust as the issue of reliability of an agent’s transmission. They use a variant of modal logic to model various trust scenarios. They also use their language to model the concepts of deception and an entity’s trust in another entity.

Yahalom et al. [22, 23] propose a formal model for deriving new trust relationships from existing ones. In [22] rules and algorithms for obtaining public keys based on trust relationships are developed. In [23] the authors propose a model for expressing trust relations in authentication protocols, together with an algorithm for deriving trust relations from recommendations. Neither of these works define what is meant by trust. Beth et al. [3] extend the ideas presented by Yahalom et al. to include relative trust. This work proposes a method for extracting trust values based on experiences and recommendations and also a method for deriving new trust values from existing ones within a network of trust relationships. Jøsang [10, 11, 12] proposes a model for trust based on a general model for expressing relatively uncertain beliefs about the truth of statements. Trust is an opinion, which is expressed as a triple $\langle b, d, u \rangle \in [0, 1]^3$. Here b , d , and u are respectively measures of one’s belief, disbelief, and uncertainty in a proposition. This work is very useful and our approach is based on this

work. Cohen et al. [5] propose an alternative, more differentiated conception of trust, called Argument-based Probabilistic Trust model (APT). The most important use of APT is to chart how trust varies, from one user to another, from one decision aid to another, from one situation to another, and across phases of decision aid use.

Xiong and Liu [15] present a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feedback system. They propose three basic trust parameters – peer feedback through transactions, total number of transactions a peer performs, and credibility of the feedback sources. The authors address factors that influence peer-to-peer trust, like reputation systems and misbehavior of peers by giving false feedback. The authors also provide a trust metric for predicting a given peer’s likelihood of a successful transaction in the future. Purser [16] presents a simple, graphical approach to model trust. He formalizes the relationship between trust and risk and argues that for every trust relationship, there exists a risk associated with a breach of the trust. Trust relationships are modeled as directed graphs where trust is an unidirectional directed edge from the trusting entity to the trusted entity. The author includes context (to define scope of trust), associated confidence level, associated risk and transitivity value. Bacharach and Gambetta [2] embark on a re-orientation of the theory of trust. They define trust as a particular belief, which arises in games with a certain payoff structure. They also identify the source of the primary trust problem in the uncertainty about the payoffs of the trustee. According to the authors, the truster must judge whether apparent signs of trustworthiness are themselves to be trusted.

The idea of trust depending on knowledge, recommendation, and experience has been proposed by Ray and Chakraborty [18, 19]. In these works, the authors argue that trust between a truster and trustee is associated with a context and this trust value depends on experience, knowledge, and recommendation. The authors propose a model in which trust is represented as a 3-element vector where the elements correspond to experience, knowledge, and recommendation. Each of these elements can either have a numeric value in the range $[-1,1]$ or have a value uncertain, denoted by \perp . The authors describe policies that can be used to determine the values corresponding to these elements. The authors also show how to normalize the trust vector to obtain a scalar value on the basis of which a user can obtain some idea about the trustworthiness of an entity or compare the trustworthiness of different entities. This model has two shortcomings. First limitation is that although the model captures the notion of uncertainty, it fails to capture the degree of uncertainty. Second limitation of the above work is that

the model relies on having experience, knowledge, and recommendation values pertaining to that context. Often times, this may not be the case – one or more values may be missing. In the worst case, if there are no experience, knowledge, and recommendation for some particular context, the model fails to find a trust value. This situation is not uncommon – often times, we have to evaluate the trustworthiness of a new entity for which very little information is available.

The above two limitations are eliminated from our current work. In this work, trust also depends on three factors, experience, knowledge, and recommendation. However, each of these has a value that is represented in the form of a triple (b, d, u) where b , d , and u denote belief, disbelief, and uncertainty as in Jøsang’s model [10, 11, 12]. This allows us to capture the degree of uncertainty. We formalize the notion of contexts and show how different contexts are related with each other. Thus, for a certain context if the truster cannot find a value corresponding to recommendation, experience, and knowledge, he can still evaluate the trust based on the values obtained from related contexts.

3 Overview of The Trust Model

We adopt the definition of trust as provided by Grandison and Sloman [6].

Definition 1 Trust is defined to be the firm belief in the competence of an entity to act dependably, reliably and securely within a specific context.

In the same work, Grandison and Sloman define *distrust* as the “lack of firm belief in the competence of an entity to act dependably, securely and reliably”. However, we believe distrust is somewhat stronger than just “lacking a belief”. Grandison and Sloman’s definition suggests the possibility of ambivalence in making a decision regarding distrust. We choose to be more precise and thus define distrust as follows.

Definition 2 Distrust is defined as the firm disbelief in the competence of an entity to act dependably, securely and reliably within a specified context.

In our model, trust is specified as a trust relationship between a truster, say entity A , – an entity that trusts the target entity – and a trustee, say entity B , – the entity that is trusted. The truster A is always an active entity (for example, a human being or a subject). The trustee B can either be an active entity or a passive entity (for example, a piece of information or a software).

The trust relationship between a truster, A , and a trustee, B , is never absolute [6]. Always, the truster trusts the trustee with respect to its ability to perform a specific action or provide a specific service. For example, an entity A may trust another entity B about the latter's ability to keep a secret. However, this does not mean if A wants a job done efficiently, A will trust B to do it. Similarly, if we want to compare two trust values, we just cannot compare two arbitrary trust values. We need to compare the values for trust which serves similar purposes. This leads us to associate a notion of *context* with a trust relationship. Examples of contexts are (i) to *provide a service*, (ii) to *make decisions* on behalf of a truster, and (iii) to *access resources* of a truster. (The formal definition of context appears in Section 5.)

We adapt Jøsang's *opinion model*, introduced in [10], in our model of trust. In his work, opinion is represented as a triple (b, d, u) where b represents *belief*, d represents *disbelief* and u represents *uncertainty*. Each of these components has a value between $[0, 1]$ and sum of the three components is 1. Thus, an opinion (b, d, u) is represented as a point in the *opinion space* which he represents by a unit equilateral triangle. We adapt this model to represent a truster A 's trust on a trustee B on some context c as a triple $({}_A b_B^c, {}_A d_B^c, {}_A u_B^c)$, where ${}_A b_B^c$ is A 's *belief* on B in context c , ${}_A d_B^c$ is A 's *disbelief* on B in context c , and ${}_A u_B^c$ is truster A 's *uncertainty* about B in the context c .

The trust relationship is not static but changes over time. Even if there is no change in the underlying factors that influence trust over a time period, the values of the trust relationship at the end of the period is not the same as that at the beginning of the period. Thus, we need to specify time when describing a trust relationship. The trust relationship between truster A and trustee B pertaining to context c at time t is formally denoted as $(A \xrightarrow{c} B)_t$.

The trust relationship $(A \xrightarrow{c} B)_t$ is a 3×3 matrix. The rows of the matrix correspond to the three parameters, namely, *experience*, *knowledge*, and *recommendation*, on which trust depends. (The formal definition of these parameters and methods for evaluating them are given in section 4.) Each of these parameters is represented in terms of (b, d, u) where b means belief on the parameter for evaluating trust, d specifies disbelief on the parameter, and u means uncertainty about the parameter to evaluate the trust. These three terms constitute the columns of the trust matrix.

The three parameters may not have equal importance for evaluating trust. The *trust policy vector* specifies the normalization factor that gives the relative weight of each parameter. Applying the normalization factor to the trust relationship gives a *normalized trust relationship*.

The normalized trust relationship between truster A and trustee B pertaining to context c at time t is formally denoted as $(A \xrightarrow{c} B)_t^N$. It specifies A 's *normalized* trust on B at a given time t for a particular context c . This normalized trust is represented as a single triple $(\hat{A}b_B^c, \hat{A}d_B^c, \hat{A}u_B^c)$.

4 Trust Evaluation

We next describe in details how trust evaluation takes place in our model. We begin by formally defining the three parameters on which trust relationship depends, namely, experience, knowledge, and recommendation.

Definition 3 The *experience* of a truster about a trustee is defined as the measure of the cumulative effect of a number of events that were encountered by the truster with respect to the trustee in a particular context and over a specified period of time.

The trust of a truster on a trustee can change because of the the truster's *experiences* with the trustee in the particular context. The experience depends on the type of events, namely, positive, negative or neutral, that have been encountered by the truster.

Definition 4 The *knowledge* of the truster regarding a trustee for a particular context is defined as a measure of the condition of awareness of the truster through acquaintance with, familiarity of or understanding of a science, art or technique.

The trust of a truster on a trustee can change because of some *knowledge* that the truster comes to possess regarding the trustee for the particular context. Knowledge can be of two types – *direct knowledge* and *indirect knowledge*. Direct knowledge is one which the truster acquires by itself. It may be obtained by the truster in some earlier time for some purpose or, it may be a piece of information about the trustee for which the truster has a concrete proof to be true. Indirect knowledge, on the other hand, is something that the truster does not acquire by itself. The source of indirect knowledge is the *reputation* of the trustee in the context. The truster may get the idea about the reputation of trustee from various sources like reviews, journals, news bulletin, people's opinion etc.

Definition 5 A *recommendation* about a trustee is defined as a measure of the subjective or objective judgment of a recommender about the trustee to the truster.

The trust value of a truster on a trustee can change because of a *recommendation* for the trustee. A recommender sends her opinion, in terms of a triple (b, d, u) , on the trustee in the specified context as the recommendation. Moreover, recommendation can be obtained by the truster from more than one source.

In the following sections, we describe how experience, knowledge, and recommendation, can be evaluated.

4.1 Evaluating Experience

We model experience in terms of the number of events encountered by a truster, A , regarding a trustee, B in the context c within a specified period of time $[t_0, t_n]$. We assume that A has a record of the events since time t_0 . An event can be positive or negative or neutral. Positive events contribute towards increasing the belief component of experience. Negative events increase the disbelief component of experience. Neutral events increase both belief and disbelief components equally. No experience contributes towards the uncertainty component of experience. In the following, we describe how to calculate the experience that a truster A has about trustee B with respect to context c . This is formally denoted as ${}^A E_B^c = (b_E, d_E, u_E)$ where b_E, d_E, u_E represent belief, disbelief and uncertainty components respectively with respect to the experience that A has towards B .

Let \mathbb{N} denote the set of natural numbers. The set of time instances $\{t_0, t_1, \dots, t_n\}$ is a totally ordered set. The ordering relation, called the *precedes-in-time* relation and denoted by \prec , is defined as follows: $\forall i, j \in \mathbb{N}, t_i \prec t_j \Leftrightarrow i < j$. We use the symbol $t_i \preceq t_j$ to signify either $t_i \prec t_j$ or $t_i = t_j$.

Consecutive time instances are grouped into an interval. We use the temporal notation $[t_i, t_j]$ for describing a time interval where $t_i \preceq t_j$. The time interval $[t_i, t_j]$ describes the set of consecutive time instances where t_i is the first instance and t_j is the last one. We denote the time period of interest as $[t_0, t_n]$. This is divided into a set of n sub-intervals $[t_0, t_1], [t_1, t_2], \dots, [t_{n-1}, t_n]$. The intervals overlap at the boundary points only. That is, $\forall i, j, k, l \in \mathbb{N}$, where i, j, k, l are all distinct, $[t_i, t_j] \cap [t_k, t_l] = \emptyset$. Also, $\forall i, j, k \in \mathbb{N}$, where i, j, k are all distinct, $[t_i, t_j] \cap [t_j, t_k] = \{t_j\}$. That is, all instances, except t_0 and t_n , that occur at the boundary of an interval is a part of two intervals. We refer to the interval $[t_{k-1}, t_k]$ as the k^{th} interval where $0 \leq k \leq n - 1$.

We assume that events occur at time instances. The function ET , referred to as the event-

occurrence-time function, returns the time instance t_j at which a given event e_k occurred. Formally, $ET(e_k) = t_j$. Moreover, if $ET(e_k) = t_j$ and $t_j \in [t_i, t_k]$ and $j \neq i \wedge j \neq k$, then e_k is said to occur in the interval $[t_i, t_k]$. For two consecutive intervals $[t_i, t_j]$ and $[t_j, t_k]$ if $ET(e_i) = t_j$ then we assume e_k occurs in the interval $[t_i, t_j]$.

Let the experience acquired at interval i , $1 \leq i \leq n-1$, be represented as (b_i, d_i, u_i) where b_i, d_i, u_i denotes belief, disbelief, and uncertainty respectively. When no event occurs during some particular time interval i , this corresponds to the fact that $u_i = 1$ and $b_i = d_i = 0$. The next case is when events occur at the interval i . Let P_i denote the set of all positive events, Q_i denote the set of all negative events, and N_i denote the set of all neutral events that occur in the interval i . Each positive event increases b_i , each negative event increases d_i , and each neutral event increase both b_i and d_i . The values for b_i, d_i and u_i are computed as follows. $b_i = \frac{|P_i| + \frac{|N_i|}{2}}{|P_i| + |Q_i| + |N_i|}$, $d_i = \frac{|Q_i| + \frac{|N_i|}{2}}{|P_i| + |Q_i| + |N_i|}$, and $u_i = 0$. The intuition is that each positive event contributes to the belief component by $\frac{1}{|P_i| + |Q_i| + |N_i|}$. Similarly, each negative event contributes to the disbelief component by $\frac{1}{|P_i| + |Q_i| + |N_i|}$. Each neutral event contributes equally to both belief and disbelief component by $\frac{0.5 * |N_i|}{|P_i| + |Q_i| + |N_i|}$. Moreover, since events have occurred in the interval, the uncertainty component is 0.

Note that, in real world, events occurring in the distant past has less effect than those that have recently occurred. More importance must be given to recent events than past ones. To accommodate this in our model, we assign a *non-negative* weight w_i to the i^{th} interval such that $w_i > w_j$ whenever $j < i$, $i, j \in \mathbb{N}$. We use the formula $w_i = \frac{i}{S} \forall i = 1, 2, \dots, n$ where $S = \frac{n(n+1)}{2}$ to evaluate weights of the intervals, satisfying the above condition.

The experience of A about B in context c is expressed as, ${}_A E_B^c = (b_E, d_E, u_E)$. The values of b_E, d_E , and u_E are given by $b_E = \sum_{i=1}^n w_i * b_i$, $d_E = \sum_{i=1}^n w_i * d_i$, and $u_E = \sum_{i=1}^n w_i * u_i$ respectively.

4.2 Evaluating Knowledge

The knowledge factor is made up of two parts: *direct knowledge* and *indirect knowledge*. Direct knowledge can be formally assessed or evaluated. Indirect knowledge is more subjective. Direct knowledge can be evaluated through credentials and certificates. Indirect knowledge can be obtained by reputation. Direct knowledge and indirect knowledge are associated with triples $K_D = (b_D, d_D, u_D)$ and $K_I = (b_I, d_I, u_I)$ respectively. Each piece of direct (indirect) knowledge is categorized into positive, negative, or neutral. The elements of the triple

(b_D, d_D, u_D) can be computed as follows. $b_D = \frac{\#positive\ direct\ knowledge + \#neutral\ direct\ knowledge/2}{total\ number\ of\ direct\ knowledge}$.
 $d_D = \frac{\#negative\ direct\ knowledge + \#neutral\ direct\ knowledge/2}{total\ number\ of\ direct\ knowledge}$. If there is any direct knowledge $u_D = 0$, otherwise $u_D = 1$. Similar formulas can be written for indirect knowledge.

The weight that a truster assigns to each of these knowledge types depends on the problem context. The truster assigns the relative weights w_D, w_I for these two types of knowledge, where $w_D, w_I \in [0, 1]$ and $w_D + w_I = 1$. The weights are determined by the underlying policy. Truster A 's knowledge about trustee B in the context c is computed as

$$\begin{aligned} {}_A K_B^c &= w_D \times K_D + w_I \times K_I \\ &= w_D \times (b_D, d_D, u_D) + w_I \times (b_I, d_I, u_I) \\ &= (b_K, d_K, u_K) \end{aligned}$$

where $b_K = w_D \times b_D + w_I \times b_I$, $d_K = w_D \times d_D + w_I \times d_I$, $u_K = w_D \times u_D + w_I \times u_I$.

4.3 Evaluating Recommendation

The truster A may obtain a recommendation from multiple recommenders regarding trustee B in the context c . The goal is to generate a triple (b, d, u) from each recommender and use these to get (b_R, d_R, u_R) which represents the recommendation that A has received about B with respect to context c . First, we give the details about how the triple is computed for each recommender. Later, we describe how these results are aggregated.

Let M be one such recommender. The recommender M may or may not have a trust relationship with trustee B regarding context c . The truster A can provide a questionnaire to the recommender. The recommender is allowed to use the values +1, -1, 0, or \perp in filling this questionnaire. The value +1 indicates belief, -1 indicates disbelief, 0 indicates neutral, and \perp indicates unknown. The number of \perp s with respect to the total number of values will give a measure of uncertainty. The ratio of the number of +1s together with half the number of 0s to the total number of values gives the value for belief. The ratio of the number of -1s together with half the number of 0s to the total number of values gives the value for disbelief. If the recommender does not return a recommendation, the truster uses the triple (0,0,1) as a recommendation from M .

The truster A will have a trust relationship with the recommender M . The context of this trust relationship will be to act "reliably to provide a service (recommendation, in this case)".

This trust relationship will affect the opinion of the recommendation provided by the recommender. The truster scales the recommender's opinion about the trustee with this trust value. Scaling the recommendation score based on the trust relationship between the truster and the recommender has one important benefit. Suppose that the recommender tells a lie about the trustee in the recommendation in order to gain an advantage with the truster. If the truster does not have belief on the recommender to a great degree then the belief on the recommendation will be low with the truster. Note also that if the truster disbelieves a recommender to properly provide a recommendation, it will most likely not ask for the recommendation.

The trust relationship that truster A has with trustee M in the context of providing a recommendation is represented as a 3×3 matrix. The rows of the matrix correspond to experience, knowledge, and recommendation and the columns correspond to belief, disbelief, and uncertainty. This matrix is normalized as outlined in Section 4.4 and converted into a triple of the form (b, d, u) . This triple will be used for the scaling operation.

To do this scaling, we borrow the concept of "discounting" proposed by Jøsang [13, 14]. According to his proposition, if the recommender M disbelieves the trustee B or is uncertain about B , then A also disbelieves B or is uncertain about B to the extent scaled down by A 's belief on M . Also, A 's disbelief and uncertainty about M 's opinion contribute towards A 's uncertainty about B . If M sends the triple ${}_M b_B, {}_M d_B, {}_M u_B$ as a recommendation about B , and A has the trust on M as $({}_A b_M, {}_A d_M, {}_A u_M)$, then the *recommendation* ${}_M R_B^c$ of a recommender M for an entity B to the truster A in a context c is given by $({}_A b_B^R, {}_A d_B^R, {}_A u_B^R)$. The values of ${}_A b_B^R, {}_A d_B^R, {}_A u_B^R$ computed as per Jøsang's formula is:

$$\begin{aligned} {}_A b_B^R &= {}_A b_M \times {}_M b_B \\ {}_A d_B^R &= {}_A b_M \times {}_M d_B \\ {}_A u_B^R &= {}_A d_M + {}_A u_M + {}_A b_M \times {}_M u_B \end{aligned}$$

Recall that the truster A may get recommendations about the trustee B from many different recommenders. Then A 's belief on the recommendation about B is the average of the belief values of all recommendations and A 's disbelief is the average of the disbelief values of the recommendations. The same is true for A 's uncertainty about the recommendations. Therefore, if ψ is a group of n recommenders then ${}_{A\psi} b_R = \frac{\sum_{i=1}^n {}_{A_i} b_B^R}{n}$, ${}_{A\psi} d_R = \frac{\sum_{i=1}^n {}_{A_i} d_B^R}{n}$ and ${}_{A\psi} u_R = \frac{\sum_{i=1}^n {}_{A_i} u_B^R}{n}$. Hence, the recommendation component is expressed by the triple $({}_{A\psi} b_R, {}_{A\psi} d_R, {}_{A\psi} u_R)$.

4.4 Normalization of Trust Vector

Having determined the triples for each component of trust we specify the simple trust relationship between the truster A and the trustee B in a context c at time t as

$$(A \xrightarrow{c} B)_t = \begin{pmatrix} b_E & d_E & u_E \\ b_K & d_K & u_K \\ A\psi b_R & A\psi d_R & A\psi u_R \end{pmatrix} \quad (1)$$

Given the same set of values for the factors that influence trust, two trusters may come up with two different trust for the same trustee. Grandison [6] refers to this characteristic as propensity to trust. This may happen because a truster may assign different weights to the different factors that influence trust. A truster may give more weight to one of the parameters than another in computing a trust relationship. For example, a truster A may choose to lay more emphasis on experience than recommendation in computing trust. Alternatively, a truster may be quite skeptical regarding a recommendation about the trustee. In that case, the truster may want to consider the recommendation factor to a lesser extent in computing trust than experience and knowledge about the trustee. Which particular component needs to be emphasized more than the others, is a matter of trust evaluation policy of the truster. The policy is represented by the truster as a trust policy vector.

Definition 6 The *trust policy vector*, ${}_A W_B^c$, is a vector that has the same number of components as the simple-trust vector. The elements are real numbers in the range $[0, 1]$ and the sum of all elements is equal to 1.

The elements of this vector are weights corresponding to the parameters of trust relationship. Let $(A \xrightarrow{c} B)_t$ be the simple trust relationship between truster A and trustee B in context c at time t . Let also ${}_A W_B^c = [W_E, W_K, W_R]$ be the corresponding trust evaluation policy vector elements such that $W_E + W_K + W_R = 1$ and $W_E, W_K, W_R \in [0, 1]$. Therefore, the normalized trust relationship between a truster A and a trustee B at a time t and for a particular context c is given

by

$$\begin{aligned}
(A \xrightarrow{c} B)_t^N &= {}_A W_B^c \times (A \xrightarrow{c} B)_t \\
&= (W_E, W_K, W_R) \times \begin{pmatrix} b_E & d_E & u_E \\ b_K & d_K & u_K \\ {}_{A\Psi} b_R & {}_{A\Psi} d_R & {}_{A\Psi} u_R \end{pmatrix} \\
&= ({}_A \hat{b}_B^c, {}_A \hat{d}_B^c, {}_A \hat{u}_B^c)
\end{aligned}$$

where ${}_A \hat{b}_B^c = W_E \times b_E + W_K \times b_K + W_R \times {}_{A\Psi} b_R$, ${}_A \hat{d}_B^c = W_E \times d_E + W_K \times d_K + W_R \times {}_{A\Psi} d_R$, ${}_A \hat{u}_B^c = W_E \times u_E + W_K \times u_K + W_R \times {}_{A\Psi} u_R$.

It follows from above that each element ${}_A \hat{b}_B^c, {}_A \hat{d}_B^c, {}_A \hat{u}_B^c$ of the normalized trust relationship lies within $[0, 1]$ and ${}_A \hat{b}_B^c + {}_A \hat{d}_B^c + {}_A \hat{u}_B^c = 1$.

4.5 Trust Dynamics

Belief, disbelief, and uncertainty changes over time. Thus, trust also changes over time. Let us suppose that we have initially computed a trust relationship T_{t_i} at time t_i , based on the values of the belief, disbelief, and uncertainty component of underlying parameters at that time. Suppose now that we try to recompute the trust relationship T_{t_n} at time t_n . We claim that even if the underlying parameters do not change between times t_i and t_n , the belief, disbelief, and uncertainty of the trust relationship will change, thereby changing the trust relationship. This change of trust over time is often called *trust dynamics*.

To model trust dynamics we refer to the old adage – time the great healer. The general tendency is to forget about past happenings. This leads us to claim that belief (and disbelief) tends towards zero as time increases, thereby increasing uncertainty. Initially, the value does not change much; after a certain period the change is more rapid; finally the change becomes more stable as the value approaches the 0 level. Also we assert that $\lim_{t \rightarrow \infty} b = 0$ and $\lim_{t \rightarrow \infty} d = 0$.

How fast belief (or disbelief) will decay over time, is, we believe, dependent on the truster's policy. The truster may choose to forget about his belief (or disbelief) which are 3 years old or 5 years old. The model cannot dictate this. Our goal is to provide a basis by which the truster can at least estimate, based on the truster's individual perception about this, the trust at time t_n . We further believe that trust relationship at present time is not only dependent on the values of the underlying parameters, but also on the “decayed” values of the previous trust. We discuss

this in more details in the next section.

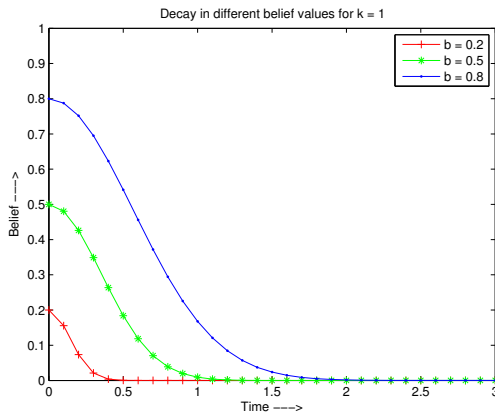
Let b_{t_i} , be the value of ‘belief’ component of a trust relationship T_{t_i} at time t_i , and b_{t_n} be the decayed value of the same at time t_n . Then the *time-dependent value* of b_{t_i} is defined as follows:

Definition 7 The *time-dependent value* of belief of a trust relationship T_{t_i} from time t_i , computed at present time t_n , is given by

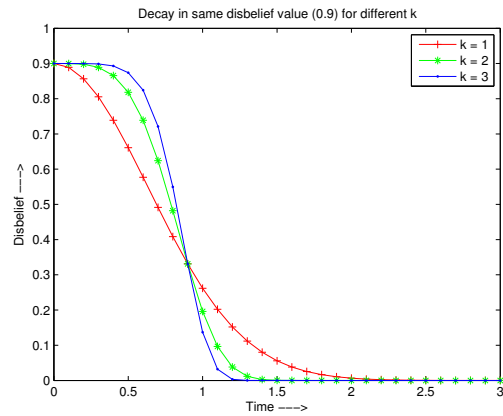
$$b_{t_n} = b_{t_i} \times e^{-((b_{t_i})^{-1}\Delta t)^{2k}} \quad (2)$$

where $\Delta t = t_n - t_i$ and k is any small integer ≥ 1 .

Similarly, we define time-dependent value of disbelief. Hence, the time-dependent value of uncertainty is obtained as $u_{t_n} = 1 - b_{t_n} - d_{t_n}$. This equation shows that, as belief and disbelief decrease over time, uncertainty increases. Figure 1(a) shows the nature of decay in different belief values with the same decay rate $k = 1$ and figure 1(b) shows the nature of decay of disbelief value 0.9 with different k .



(a) Decay in different belief values for same $k = 1$



(b) Decay in disbelief value 0.9 for different k

The value of k determines the rate of change of belief (or, disbelief) with time and is assigned by the truster based on its perception about the change. If $\Delta t = 0$ that is at $t_n = t_i$, $e^{-((b_{t_i})^{-1}\Delta t)^{2k}} = 1$ and hence $b_{t_n} = b_{t_i}$. When $\Delta t \rightarrow \infty$, then $e^{-((b_{t_i})^{-1}\Delta t)^{2k}} \rightarrow 0$ and hence

¹We would like to thank Michael Stevens and Paul D. Williams [20] for suggesting a better modification of the equation for time dependent value.

$b_{t_n} \rightarrow 0$. This corroborates the fact the time-dependent value of the last known belief (or, disbelief) value is asymptotic to zero at infinite time. Thus, the trust relationship T_{t_n} at time t_n is specified as $b_{t_n}, d_{t_n}, u_{t_n}$.

Note, it is not necessary to have the same decay rate for belief and disbelief. A truster may choose to have two different values k and k' for belief and disbelief respectively.

4.6 Trust Vector at Present Time

As indicated earlier, the trust of a truster A on a trustee B in a context c at time t_n depends not only on the underlying components of the trust relationship but also on the trust established earlier at time t_i . Consider for example that at time t_i Alice trusts Bob to the fullest extent $(1, 0, 0)$. At time t_n Alice re-evaluates the trust relationship and determines the value to be $(0.2, 0.5, 0.3)$. However, we believe that Alice will lay some importance to the previous trust and will not distrust Bob as much as the new trust relationship. So, the normalized trust vector at t_n is a linear combination of time-dependent trust and the normalized trust calculated at present time. The weight Alice will give to old trust vector and present normalized trust vector is, again, a matter of policy. However, this leads us to refine the expression for normalized trust vector at time t_n as follows. This refinement is given by the following definition where α and β are the weights corresponding to present normalized trust vector $({}_A\hat{b}_B^c, {}_A\hat{d}_B^c, {}_A\hat{u}_B^c)$ and time-dependent trust vector $(b_{t_n}, d_{t_n}, u_{t_n})$, respectively.

Definition 8 The normalized trust relationship between a truster A and a trustee B at time t_n in a particular context c taking into account the old trust values is given by

$$\begin{aligned} (A \xrightarrow{c} B)_{t_n}^N &= \alpha \times ({}_A\hat{b}_B^c, {}_A\hat{d}_B^c, {}_A\hat{u}_B^c) + \beta \times (b_{t_n}, d_{t_n}, u_{t_n}) \\ &= (\alpha \times {}_A\hat{b}_B^c + \beta \times b_{t_n}, \alpha \times {}_A\hat{d}_B^c + \beta \times d_{t_n}, \alpha \times {}_A\hat{u}_B^c + \beta \times u_{t_n}) \\ &= ({}_Ab_B^c, {}_Ad_B^c, {}_Au_B^c) \end{aligned}$$

where ${}_Ab_B^c = \alpha \times {}_A\hat{b}_B^c + \beta \times b_{t_n}$, ${}_Ad_B^c = \alpha \times {}_A\hat{d}_B^c + \beta \times d_{t_n}$, ${}_Au_B^c = \alpha \times {}_A\hat{u}_B^c + \beta \times u_{t_n}$ and $\alpha + \beta = 1$, $\alpha, \beta \in [0, 1]$.

4.7 Comparison Operation on Trust Vectors

In many real life scenarios we need to determine the relative trustworthiness of two trustees. Consider the following example. Suppose entity A gets two conflicting pieces of information

from two different sources B and C . In this case A will probably want to compare its trust relationships with entities B and C and accept the information that originated from the more trustworthy entity. The concept of more trustworthy is captured using dominance relation.

Definition 9 Consider two trust relationship $T = (b_T, d_T, u_T)$ and $T' = (b_{T'}, d_{T'}, u_{T'})$ that have been defined over the same context and using the same policy vector. We say T dominates T' , denoted by $T \succ T'$, if any of the following conditions hold.

1. $b_T > b_{T'}$
2. $b_T = b_{T'} \wedge d_T < d_{T'}$

The dominance relation compares the two trust relationships using the belief component. The relationship with the higher belief value is said to dominate the other. When the two relationships have identical beliefs, we compare them on the basis of disbelief. Since there is no well-defined interpretation for uncertainty, we do not use it for computing dominance relation. The formula given above compares trust relationships only when contexts are identical. In the next section, we show how to overcome this restriction.

5 Reasoning about Trust Relationships in Different Contexts

The model we have described so far has two shortcomings that needs to be overcome if the model is to be useful for real-world applications. First, it is not possible to compute a useful trust vector if the truster does not have any experience, knowledge, or recommendation about a trustee in a given context. The model returns the vector $(0, 0, 1)$ – total uncertainty. Second, the model developed so far can reason about trust relationships only with respect to a given context. In other words, it allows trust vectors to be compared only when there is an exact match on the context. For this to happen the contexts needs to be specified using exactly the same terms. This assumption is not realistic in most situations. It is extremely unlikely that different trusters will specify a given context in exactly the same manner. This prevents our model from being interoperable. However, it appears that a model providing such features will be useful. For example, let a user A (the truster) trust a software developer B (the trustee) to a degree T to produce excellent quality anti-virus software (the context). Assuming that expertise to develop an anti-virus software (a related context) is similar to the expertise needed

to develop anti-spam software, it seems natural that the truster A will be able to determine how much to trust B for the different (but related) context. We introduce this feature in the model by formalizing a notion of context and the relationships that exist between different contexts.

We observe that we must define contexts in such a manner that makes our model interoperable. Different entities often use different words to describe the same context. Alternately, the same word can be used for describing different contexts. These are example of semantic conflicts in the use of terminology. To solve these problems we borrow some ideas from the work on ontologies [7, 21]. Our ontology consists of a set of contexts together with relationships defined among them. We begin by giving a formal definition of context and later describe the relationships between contexts.

Definition 10 [Context] A *context* C_i is represented by a set of keywords that is denoted by $KeywordSet_{C_i}$.

Each keyword in $KeywordSet_{C_i}$ is used to describe the context C_i . The keywords in $KeywordSet_{C_i}$ are semantically equivalent because they express the same context. For each context C , we require that the $KeywordSet_C$ should be non-empty and finite. For any two distinct contexts C and C' , $KeywordSet_C \cap KeywordSet_{C'} = \{\}$. In other words, any keyword belongs to exactly one context. An example will help illustrate the notion of contexts. The context *age* can be expressed by the keywords $\{age, yearOfBirth\}$.

Consider the two contexts *doing a job* and *doing a job well*. Modeling them as distinct concepts increases the total number of contexts that must be managed. To solve this problem, we specify *doing a job* as a context and associate a set of values with it. The values in this case will be $\{badly, neutral, well\}$. Using these values, we can specify different conditions on the context. Each of these conditions represent a *derived context*. To obtain a derived context from the context C_i , each keyword k , where $k \in KeywordSet_{C_i}$, must be associated with a domain D_k that defines the set of values associated with the keyword. The formal definition of derived context appears below.

Definition 11 [Derived Context] A *derived context* $\mathcal{D}C_i$ is one that is specified by a condition $k \text{ op } v$ defined over a context C_i where $k \in KeywordSet_{C_i}$ and $v \in D_k$ and op is a logical operator compatible with the domain of D_k .

To check whether two derived contexts specified using conditions on different keywords are equivalent, we need the notion of translation functions.

Definition 12 [Translation Function] The *translation function* associated with a context C_i , denoted as TF_{C_i} , is a total function that takes as input a condition $k \text{ op } v$ ($k \in \text{KeywordSet}_{C_i}$) and a keyword k' ($k' \in \text{KeywordSet}_{C_i}$) and produces an equivalent condition defined over keyword k' . This is formally expressed as follows. $TF_{C_i} : \text{Cond}_{C_i} \times \text{KeywordSet}_{C_i} \rightarrow \text{Cond}_{C_i}$ where Cond_{C_i} is the set of all valid conditions specified over the keywords in KeywordSet_{C_i} .

Since the translation function is total, for every given valid condition and keyword there exists an equivalent condition defined on the given keyword. Several steps are involved in developing the translation function. To express $k \text{ op } v$ in terms of k' , we need to first convert the value k to an equivalent value that is in the domain of k' . This step is performed by conversion functions which convert the value of one keyword to an equivalent value of another keyword. The second step is to convert the operator op into an equivalent operator op' that is suitable for the domain of k' . The definition of the conversion function together with the domain of the keyword can determine how the operator must be changed. Consider the two keywords *age* and *yearOfBirth*. Suppose we want to translate $age > 18$ to an equivalent condition defined over *yearOfBirth*. The first step is to convert $age = 18$ to an equivalent value defined over *yearOfBirth*. The function that converts *age* to *yearOfBirth* will be specified as: $yearOfBirth = \text{currentYear} - age$. For $age = 18$, this function returns $yearOfBirth = 1987$. Since *yearOfBirth* and *age* are inversely related, (that is, *age* increases as *yearOfBirth* decreases) the operator $>$ is inverted to obtain $<$. The results obtained by the TF_{C_i} function in this case will be $yearOfBirth < 1987$.

5.1 Relationships between Contexts

We now describe two kinds of relations that may exist between distinct contexts. One is the generalization/specialization relationship existing between related contexts. The other is the composition relationship between possibly unrelated contexts.

5.1.1 Specialization Relation

Distinct contexts may be related by the specialization relationship. The specialization relation is anti-symmetric and transitive. We use the notation $C_i \subset C_j$ to indicate that the context C_i is a generalization of context C_j . Alternately, context C_j is referred to as the specialization of context C_i . For instance, the contexts *makes decision* and *makes financial decisions* are

related by the specialization relationship, that is, *makes decisions* \subset *makes financial decisions*. Also, *makes financial decisions* \subset *makes payment decisions*. By transitivity, *makes decisions* \subset *makes payment decisions*.

Each specialization relationship is associated with a degree of specialization. This indicates the closeness of the two concepts. For instance, *makes payment decisions* is a specialization of *makes decision*, and *makes payment decisions* is also a specialization of *makes financial decisions*. However, the degree of specialization is different in the two cases. *makes payment decision* is closer to *makes financial decision* than *makes decision*. The *degree of specialization* captures this difference. Since two contexts related by specialization will not be exactly identical, the degree of specialization will be denoted as a fraction. The exact value of the fraction will be determined using domain knowledge.

The specialization relationship will be used in trust evaluation when information cannot be obtained for a particular context, and the values obtained from the generalized or specialized context will need to be extrapolated.

5.1.2 Composition Relation

Specialization captures the relationship between contexts that are related. Sometimes unrelated contexts can be linked together using the composition relation. We now describe this composition relation. A context in our model can either be an *elementary* context or a *composite* context. An elementary context is one which cannot be subdivided into other contexts. A composite context is one that is composed from other contexts using the logical and operation. The individual contexts that form a composite contexts are referred to as the *component* contexts. A component context can either be composite or elementary.

We use the notation $C_i \ll C_j$ to indicate that the context C_i is a component of context C_j . In such cases, C_i is referred to as the component context and C_j is the composite context. For instance, we may have the component contexts *secure key generation* and *secure key distribution* that can be combined to form the composite context *secure key generation and distribution*. This is denoted as *secure key generation* \ll *secure key generation and distribution*.

Sometimes a composite context C_i may be composed from the individual contexts C_j , C_k and C_m . All these contexts may not contribute equally to form C_i . The *degree of composition* captures this idea. A degree of composition is associated with each composition relation. Since two contexts related by composition will not be exactly identical, the degree of composition is

denoted as a fraction. The sum of all these fractions equals one if C_i is composed of C_j , C_k , and C_m only. If C_i is composed of C_j , C_k , and C_m and also other component contexts, then the sum of fractions associated with C_j , C_k , and C_m must be equal to or less than one. The exact value of the fraction representing the degree of composition will be determined by domain knowledge.

The composition relationship is important. When trust information cannot be computed for the composite context because of missing recommendation, experience, or knowledge vectors, the related information obtained from the components can be used to compute the trust vector. Alternately, if we cannot calculate the trust vector for a component context, we can use the trust vector for the composite context and extrapolate it. Later, we show how we do this.

5.1.3 Context Graphs

The specialization and the composition relations can be described using one single graph which we refer to as the *context graph*. Each node n_i in this graph corresponds to a context. There are two kinds of weighted edges in this graph: composition edges and specialization edges. A composition edge (n_i, n_j) , denoted by a solid arrow from node n_i to node n_j , indicates that the context represented by node n_i is a component of the context represented by node n_j . The weight on this edge indicates what percentage of the component context comprises the composite context. A specialization edge (n_p, n_q) , shown by a dashed arrow from node n_p to node n_q , indicates that the context represented by node n_p is a specialization of the context represented by node n_q . The weight on the edge indicates the degree of specialization of a context.

Unrelated contexts correspond to nodes in different context graphs. Each context corresponds to only one node in the set of context graphs. We denote the context graph associated with context C_i as CG_{C_i} . The formal definition of a context graph is as follows.

Definition 13 [Context Graph]

A context graph $CG = \langle \mathcal{N}, \mathcal{E}_c \cup \mathcal{E}_s \rangle$ is a weighted directed acyclic graph satisfying the following conditions.

- \mathcal{N} is a set of nodes where each node n_i is associated with a context C_i and is labeled with $KeywordSet_{C_i}$. $KeywordSet_{C_i}$ is the set of keywords associated with the context C_i .
- The set of edges in the graph can be partitioned into two sets \mathcal{E}_c and \mathcal{E}_s . For each edge $(n_i, n_j) \in \mathcal{E}_c$, the context C_i corresponding to node n_i is a component of the concept C_j

corresponding to node n_j . The weight of the edge (n_i, n_j) , denoted by $w(n_i, n_j)$, indicates the percentage of component context that makes up the composite context. For each edge $(n_i, n_j) \in \mathcal{E}_s$, the concept C_i corresponding to node n_i is a specialization of concept C_j corresponding to node n_j . Here again the weight of the edge (n_i, n_j) , denoted by $w(n_i, n_j)$, indicates the degree of specialization.

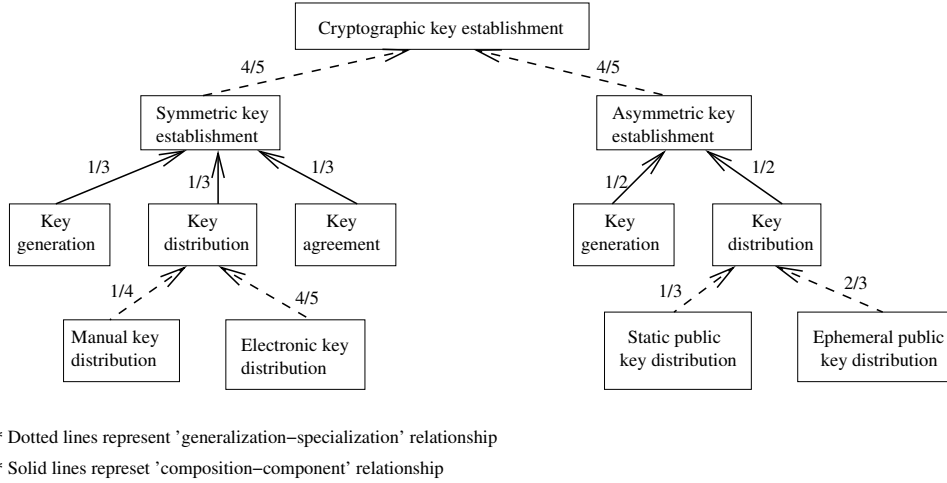


Figure 1: Specialization & Composition relationships

Figure 1 gives an example of a context graph that is associated with the context *cryptographic key establishment*. The solid arrows in this graph indicate composition relationships and the dashed arrows indicate generalization/specialization relationships. The context *cryptographic key establishment* can have two specializations, namely, *symmetric key establishment* and *asymmetric key establishment*. The weight on the edge connecting this *symmetric key establishment* with *cryptographic key establishment* indicates the degree of specialization. For instance, if symmetric key establishment is very closely related to key establishment, the degree of specialization may be labeled as $\frac{4}{5}$. Similarly, the edge connecting *asymmetric key establishment* to *key establishment* may be labeled as $\frac{4}{5}$. Each of these specific contexts is a composition of some component contexts. *Generation and distribution of symmetric keys* has three components – *key generation*, *key distribution*, and *key agreement*. A weight of $\frac{1}{3}$ can be assigned to each of these components contexts. Similarly, *generation and distribution of asymmetric keys* can have components *key generation* and *key distribution* with weights $\frac{1}{2}$ each.

A component context can also be a generalization of some specialized contexts. In the above example the context *key distribution* has two categories – *manual key distribution* and

electronic key distribution. Similarly *key distribution* in asymmetric keys can be thought of as generalization of *static public key distribution* and *ephemeral public key distribution*.

5.2 Computing the Degree of Specialization and Composition

Consider two contexts C_i and C_j where $C_i \subset C_j$, that is, C_j is a specialization of C_i . The degree of specialization is computed as follows. Let n_i, n_j be the nodes corresponding to contexts C_i and C_j in the weighted graph. Let the path from n_i to n_j consisting of specialization edges be denoted as $(n_i, n_{i+1}, n_{i+2}, \dots, n_{j-1}, n_j)$. The degree of specialization = $\prod_{p=i}^{j-1} w(n_p, n_{p+1})$. This corresponds to our notion that the similarity decreases as the length of the path from the generalized node to the specialized node increases. Note that, in real world there may be multiple paths from C_i to C_j . In such cases, it is important that the degree of specialization yield the same values when any of these paths are used for computation.

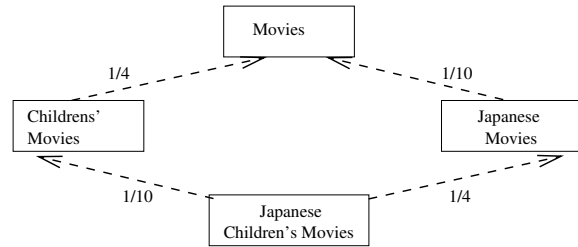


Figure 2: Computing the Degree of Specialization

An example will help illustrate this point. Consider the following specialization relationships: (a) $Movies \subset Japanese\ Movies \subset Japanese\ Children's\ Movies$ and (b) $Movies \subset Children's\ Movies \subset Japanese\ Children's\ Movies$. Suppose one is computing the degree of specialization that exists between $Movies$ and $Japanese\ Children's\ Movies$. In this case, there are two paths consisting of specialization edges between the concepts $Movies$ and $Japanese\ Children's\ Movies$. Computing the degree of specialization using any of these two paths should yield the same value. Calculating the degree of specialization that exists between $Movies$ and $Japanese\ Children's\ Movies$ using the values given in figure 2 yields $\frac{1}{40}$.

Consider two contexts C_i and C_j such that C_j is a component of C_i . Degree of composition captures what portion of C_i is made up of C_j . The degree of composition is computed as follows. Let n_i, n_j be the nodes corresponding to contexts C_i and C_j in the context graph. Let there be m paths consisting of composition edges from n_i to n_j . Let the q th path ($1 \leq$

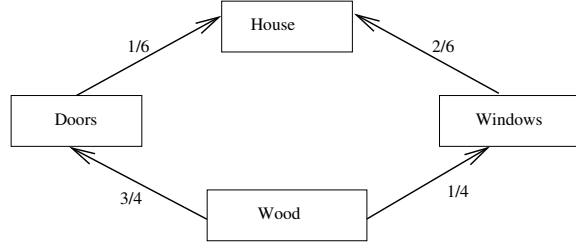


Figure 3: Computing the Degree of Composition

$q \leq m$) from n_i to n_j be denoted as $(n_i, n_{i_q+1}, n_{i_q+2}, \dots, n_{j_q-1}, n_j)$. The degree of composition $= \sum_{q=1}^m (w(n_i, n_{i_q+1}) \times w(n_{j_q-1}, n_j) \times \prod_{p=i_q+1}^{j_q-2} w(n_p, n_{p+1}))$.

A *House* is composed of *Doors*, *Windows*, and *Walls*. A *Door* is composed of *Wood*. A *Window* is composed of *Wood* and *Glass*. Therefore, we have the following composition relationships: *Windows* \ll *House*, *Doors* \ll *House*, *Wood* \ll *Windows*, and *Wood* \ll *Doors*. The composition relationships are shown in figure 3. Thus, to evaluate what part of house is made of wood, we have to consider all the paths. The degree of composition of *Wood* and *House* is $\frac{5}{24}$.

A context may be related to several other contexts through specialization and composition relationships. However, we need to find out which context or set of contexts is conceptually closest to the given context. The closest context is a singleton set if the context is a generalization or specialization of context c . It is also a singleton set if it is a composite context in which c is a component. However, if c is a composite context, then the closest concept can also be a set that contains the components of c . The formal definition appears below.

Definition 14 [Closest Context] Let c be a context. The set of contexts $S = \{c_1, c_2, \dots, c_n\}$ is defined to be *closest* to c if the following relation holds:

Case 1 – The elements in S are the components of c :

for all contexts n_i that are specializations of c

$$\text{degree of specialization}(n_i, c) \leq \sum_{j=1}^n \text{degree of composition}(c_j, c)$$

for all contexts n_i that are generalizations of c

$$\text{degree of specialization}(c, n_i) \leq \sum_{j=1}^n \text{degree of composition}(c_j, c)$$

for all composite contexts n_i in which c is a component

$$\text{degree of composition}(c, n_i) \leq \sum_{j=1}^n \text{degree of composition}(c_j, c)$$

Case 2 – S is a singleton set containing c_1 and c is a component of c_1 :

for all contexts n_i that are specializations of c

$$\text{degree of specialization}(n_i, c) \leq \text{degree of composition}(c, c_1)$$

for all contexts n_i that are generalizations of c

$$\text{degree of specialization}(c, n_i) \leq \text{degree of composition}(c, c_1)$$

for all contexts n_1, n_2, \dots, n_m that are components of c

$$\sum_{i=1}^m \text{degree of composition}(n_i, c) \leq \text{degree of composition}(c, c_1)$$

for all composite contexts n_i in which c is a component

$$\text{degree of composition}(c, n_i) \leq \text{degree of composition}(c, c_1)$$

Case 3 – S is a singleton set containing c_1 and c is a specialization of c_1 :

for all component contexts n_1, n_2, \dots, n_m of c

$$\sum_{i=1}^m \text{degree of composition}(n_i, c) \leq \text{degree of specialization}(c, c_1)$$

for all contexts n_i that are specializations of c

$$\text{degree of specialization}(n_i, c) \leq \text{degree of specialization}(c, c_1)$$

for all contexts n_i that are generalizations of c

$$\text{degree of specialization}(c, n_i) \leq \text{degree of specialization}(c, c_1)$$

for all composite contexts n_i in which c is a component

$$\text{degree of composition}(c, n_i) \leq \text{degree of specialization}(c, c_1)$$

Case 4 – S is a singleton set containing c_1 and c is a generalization of c_1 :

for all component contexts n_1, n_2, \dots, n_m of c

$$\sum_{i=1}^m \text{degree of composition}(n_i, c) \leq \text{degree of specialization}(c_1, c)$$

for all composite contexts n_i in which c is a component

$$\text{degree of composition}(c, n_i) \leq \text{degree of specialization}(c_1, c)$$

for all contexts n_i that are specializations of c

$$\text{degree of specialization}(n_i, c) \leq \text{degree of specialization}(c_1, c)$$

for all contexts n_i that are generalizations of c

$$\text{degree of specialization}(c, n_i) \leq \text{degree of specialization}(c_1, c)$$

5.3 Relationships between Context Graphs

Different information sources may use different context graphs. Comparing information or combining information that uses different context graphs may not give correct results. Before

proceeding with the comparison of information obtained from different sources, the context graphs of these sources must be merged. Note that, sometimes context graphs cannot be merged because they contain conflicting information. To understand why this happens, we first need to elaborate on the relationships that can exist between a pair of context graphs. Two context graphs can be related by any of the following relationships: (i) equality, (ii) unrelated, (iii) subsumes, and (iv) incomparable.

Definition 15 [Equality of Context Graphs] Two context graphs $CG_1 = \langle \mathcal{N}_1, \mathcal{E}_{1c} \cup \mathcal{E}_{1s} \rangle$ and $CG_2 = \langle \mathcal{N}_2, \mathcal{E}_{2c} \cup \mathcal{E}_{2s} \rangle$ are said to be equal if

1. $\mathcal{N}_1 = \mathcal{N}_2$, $\mathcal{E}_{1c} = \mathcal{E}_{2c}$ and $\mathcal{E}_{1s} = \mathcal{E}_{2s}$
2. for each $(n_i, n_j) \in (\mathcal{E}_{1c} \cup \mathcal{E}_{1s}) \cap (\mathcal{E}_{2c} \cup \mathcal{E}_{2s})$, $w_1(n_i, n_j) = w_2(n_i, n_j)$ where $w_1(n_i, n_j)$, $w_2(n_i, n_j)$ denote the weight of the edge (n_i, n_j) in graph CG_1 , CG_2 respectively.

Intuitively, two context graphs are equal if they have the same set of nodes, composition edges, and specialization edges. Moreover, each of these edges must have identical weights in the two graphs. The information obtained from identical context graphs can be compared.

Sometimes two context graphs are unrelated. They do not have any common context. It is conceivable that these graphs will be used for different situations.

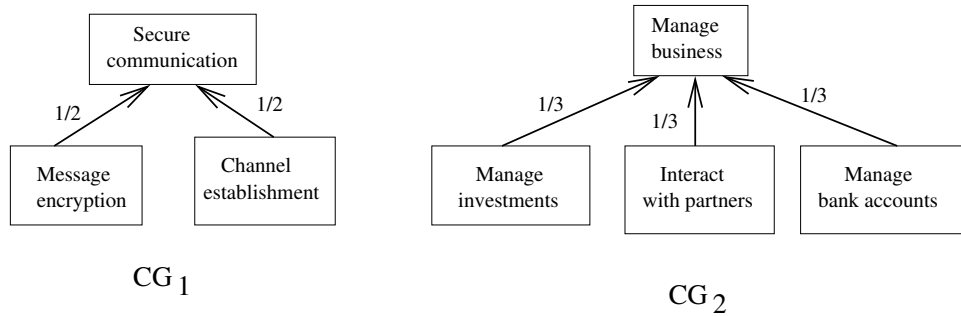


Figure 4: Unrelated Context Graphs

Definition 16 [Unrelated Context Graphs] Two context graphs CG_1 and CG_2 are said to be unrelated if $KeywordSet_{C_1} \cap KeywordSet_{C_2} = \{\}$ where $KeywordSet_{C_1}$ and $KeywordSet_{C_2}$ are the set of keywords associated with all the contexts in the context graphs CG_1 and CG_2 respectively.

Often times two context graphs are comparable but one has more information than the other. In such cases, the context graphs are related by the subsumes relation. The intuition

is that the context graph \mathcal{CG} has more information than the one it subsumes. The formal definition is given below. If \mathcal{CG}_1 subsumes \mathcal{CG}_2 , the first condition requires that the set of nodes in \mathcal{CG}_1 is greater than or equal to the set of nodes in \mathcal{CG}_2 . The second condition requires that for every specialization edge (n_i, n_j) present in \mathcal{E}_{2s} , there exists a path from n_i to n_j in \mathcal{CG}_1 consisting of specialization edges such that the product of the weight of these edges equals the weight of (n_i, n_j) in \mathcal{CG}_2 . The third condition imposes a similar requirement for the composition edges.

Definition 17 [Context Graphs related by the Subsumes Relation] Consider two context graphs $\mathcal{CG}_1 = \langle \mathcal{N}_1, \mathcal{E}_{1c} \cup \mathcal{E}_{1s} \rangle$ and $\mathcal{CG}_2 = \langle \mathcal{N}_2, \mathcal{E}_{2c} \cup \mathcal{E}_{2s} \rangle$. Let $w_1(n_i, n_j)$, $w_2(n_i, n_j)$ represent the weight of edge (n_i, n_j) in graph \mathcal{CG}_1 and \mathcal{CG}_2 respectively. \mathcal{CG}_1 is said to subsume \mathcal{CG}_2 if it satisfies all the following conditions:

1. $\mathcal{N}_2 \subseteq \mathcal{N}_1$
2. for each specialization edge $(n_i, n_j) \in \mathcal{E}_{2s}$ there exists a path $\{n_i, n_{i+1}, n_{i+2}, \dots, n_{j-1}, n_j\}$ in \mathcal{CG}_1 whose *length* ≥ 1 such that $\{(n_i, n_{i+1}), (n_{i+1}, n_{i+2}), (n_{i+2}, n_{i+3}), \dots, (n_{j-1}, n_j)\} \subseteq \mathcal{E}_{1s}$ and $\prod_{p=i}^{j-1} w_1(n_p, n_{p+1}) = w_2(n_i, n_j)$
3. for each composition edge $(n_i, n_j) \in \mathcal{E}_{2c}$ there exists m paths consisting only of composition edges where $m > 1$ in \mathcal{CG}_1 . Let the q th path ($1 \leq q \leq m$) in \mathcal{CG}_1 from n_i to n_j be denoted as $(n_i, n_{i_q+1}, n_{i_q+2}, \dots, n_{j_q-1}, n_j)$ where $\{(n_i, n_{i_q+1}), (n_{i_q+1}, n_{i_q+2}), \dots, (n_{j_q-1}, n_j)\} \subseteq \mathcal{E}_{1c}$. $w_2(n_i, n_j) = \sum_{q=1}^m (w_1(n_i, n_{i_q+1}) \times w_1(n_{j_q-1}, n_j) \times \prod_{p=i_q+1}^{j_q-2} w_1(n_p, n_{p+1}))$

Often times two context graphs, neither of which subsumes the other, may be comparable. Such graphs contain different but related information. Moreover, they never have any conflicting information. Such graphs can be merged without human intervention. Two context graphs are comparable if they satisfy a set of conditions. The first condition requires that the two graphs have one or more common nodes. The second condition requires that for any specialization edge (n_i, n_j) present in \mathcal{E}_1 , either there must be a path from n_i to n_j in \mathcal{E}_2 consisting of specialization edges in \mathcal{E}_1 and whose product equals $w_1(n_i, n_j)$ or no path exists between n_i and n_j in \mathcal{E}_2 . The third condition imposes a similar requirement for composition edges in \mathcal{E}_1 . This condition requires that either there are m paths ($m > 1$) consisting of composition edges from n_i to n_j in \mathcal{CG}_2 or there are no paths. Computing the degree of composition along these paths gives the same result as $w_1(n_i, n_j)$. The fourth and the fifth requirements imposes similar requirements for edges in \mathcal{E}_2 . The formal definition appears below.

Definition 18 [Comparable Context Graphs] Two context graphs $\mathcal{CG}_1 = \langle \mathcal{N}_1, \mathcal{E}_{1c} \cup \mathcal{E}_{1s} \rangle$ and $\mathcal{CG}_2 = \langle \mathcal{N}_2, \mathcal{E}_{2c} \cup \mathcal{E}_{2s} \rangle$ are said to be comparable if the following conditions hold.

1. $\mathcal{N}_1 \cap \mathcal{N}_2 \neq \{\}$
2. for each $(n_i, n_j) \in \mathcal{E}_{1s}$ either of the following conditions hold:
 - (a) $\exists n_{i+1}, n_{i+2}, \dots, n_{j-1} \in \mathcal{N}_2 - \mathcal{N}_1 \bullet (\{(n_i, n_{i+1}), (n_{i+1}, n_{i+2}), \dots, (n_{j-1}, n_j)\} \subseteq \mathcal{E}_{2s})$
 $\wedge \prod_{t=i}^{j-1} w_2(n_t, n_{t+1}) = w_1(n_i, n_j)$
 - (b) there does not exist any path from n_i to n_j in \mathcal{CG}_2
3. for each $(n_i, n_j) \in \mathcal{E}_{1c}$ either of the following conditions hold:
 - (a) there exists m paths ($m > 1$) from n_i to n_j in \mathcal{CG}_2 . Let the q th path ($1 \leq q \leq m$) from n_i to n_j be denoted as $(n_i, n_{i_q+1}, n_{i_q+2}, \dots, n_{j_q-1}, n_j)$ where $\{(n_i, n_{i_q+1}), (n_{i_q+1}, n_{i_q+2}), \dots, (n_{j_q-2}, n_{j_q-1}), (n_{j_q-1}, n_j)\} \subseteq \mathcal{E}_{2c}$. In such a case, $w_1(n_i, n_j) = \sum_{q=1}^m (w_2(n_i, n_{i_q+1}) \times w_2(n_{j_q-1}, n_j) \times \prod_{p=i_q+1}^{j_q-2} w_2(n_p, n_{p+1}))$
 - (b) there does not exist any path from n_i to n_j in \mathcal{CG}_2
4. for each $(n_i, n_j) \in \mathcal{E}_{2s}$ either of the following conditions hold:
 - (a) $\exists n_{i+1}, n_{i+2}, \dots, n_{j-1} \in \mathcal{N}_1 - \mathcal{N}_2 \bullet (\{(n_i, n_{i+1}), (n_{i+1}, n_{i+2}), (n_{i+2}, n_{i+3}), \dots, (n_{j-1}, n_j)\} \subseteq \mathcal{E}_{1s}) \wedge \prod_{t=i}^{j-1} w_1(n_t, n_{t+1}) = w_2(n_i, n_j)$
 - (b) there does not exist any path from n_i to n_j in \mathcal{E}_{1s}
5. for each $(n_i, n_j) \in \mathcal{E}_{2c}$ either of the following conditions hold:
 - (a) there exists m paths ($m > 1$) from n_i to n_j in \mathcal{CG}_1 . Let the q th path ($1 \leq q \leq m$) from n_i to n_j be denoted as $(n_i, n_{i_q+1}, n_{i_q+2}, \dots, n_{j_q-2}, n_{j_q-1}, n_j)$ where $\{(n_i, n_{i_q+1}), (n_{i_q+1}, n_{i_q+2}), \dots, (n_{j_q-2}, n_{j_q-1}), (n_{j_q-1}, n_j)\} \subseteq \mathcal{E}_{1c}$. In such a case, $w_2(n_i, n_j) = \sum_{q=1}^m (w_1(n_i, n_{i_q+1}) \times w_1(n_{j_q-1}, n_j) \times \prod_{p=i_q+1}^{j_q-2} w_1(n_p, n_{p+1}))$
 - (b) there does not exist any path from n_i to n_j in \mathcal{CG}_1

Definition 19 [Incomparable Context Graphs] Two context graphs that are not unrelated are incomparable if they are not comparable.

Incomparable graphs occur when the underlying assumptions are different. For example, one system may think that the degree of specialization of the contexts *makes financial decisions* and *makes payment decision* is 0.5, another might think this degree is 0.3. In this case, the two systems will generate context graphs in which some edge present in both the graphs will have

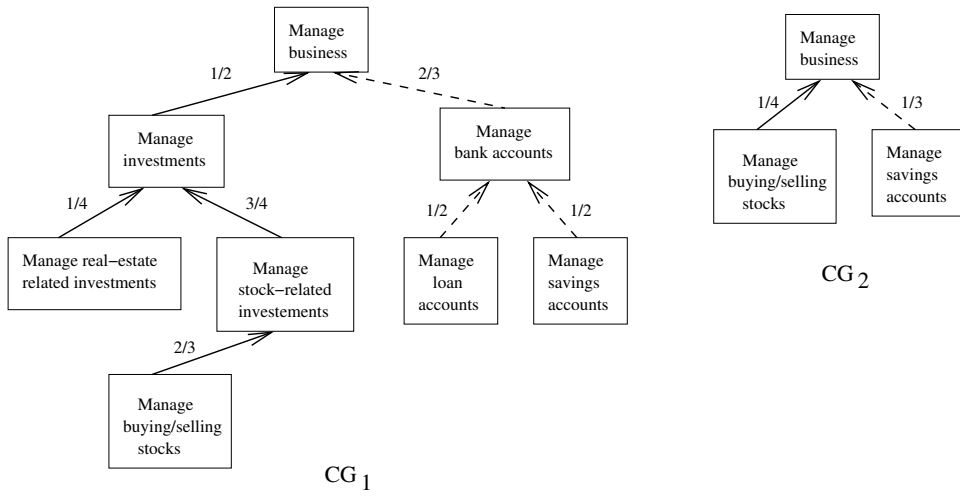


Figure 5: Context Graphs Having Subsumes Relation

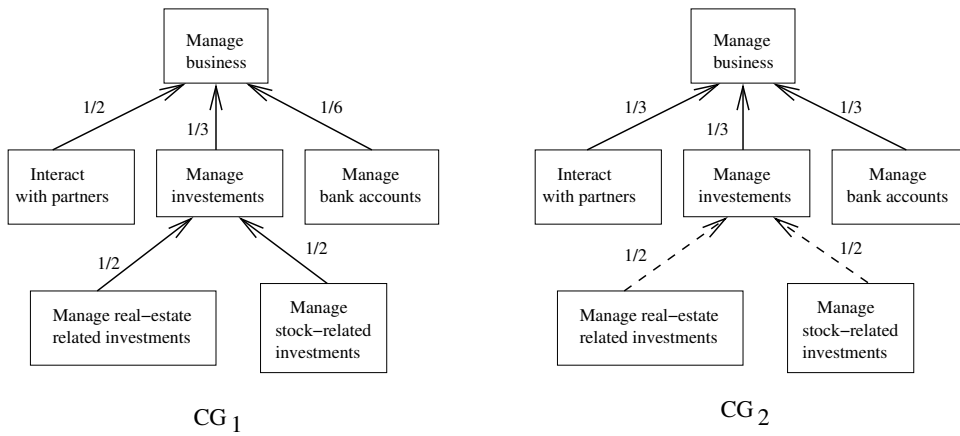


Figure 6: Incomparable Context Graphs

different weights. Alternately, one system may consider that *makes financial decisions* and *makes payment decisions* are related by generalization/specialization relationships whereas another system may consider them linked by a composition relationship. The systems will generate context graphs in which some pair of nodes present in both the graphs will be related by different types of edges. Since the conflicts are generated because of the differences in the underlying assumptions, they cannot be resolved without human intervention.

Next we give an algorithm for combining comparable graphs. The inputs to this algorithm are $CG_1 = \langle \mathcal{N}_1, \mathcal{E}_1 \rangle$ and $CG_2 = \langle \mathcal{N}_2, \mathcal{E}_2 \rangle$ – the two context graphs that must be merged. The output is $CG = \langle \mathcal{N}, \mathcal{E} \rangle$ which is the combined context graph. \mathcal{N} is computed by performing a union of the nodes in \mathcal{N}_1 and \mathcal{N}_2 . The edges common to \mathcal{E}_1 and \mathcal{E}_2 are inserted in \mathcal{E} . For each edge (n_i, n_j) that is present in \mathcal{E}_1 but not in \mathcal{E}_2 , we check if there is a path from n_i to n_j in \mathcal{E}_2 . If so, then these edges of \mathcal{E}_2 are added to \mathcal{E} . Otherwise, the edge (n_i, n_j) is added. The same process is followed for edges present in \mathcal{E}_2 but not in \mathcal{E}_1 . The resulting graph so obtained subsumes the graphs CG_1 and CG_2 .

Algorithm 1 Combining Comparable Graphs

Input: $CG_1 = \langle \mathcal{N}_1, \mathcal{E}_1 \rangle$ and $CG_2 = \langle \mathcal{N}_2, \mathcal{E}_2 \rangle$ – comparable context graphs.

Output: $CG = \langle \mathcal{N}, \mathcal{E} \rangle$ – the combined context graph.

Procedure *CombineContextGraphs*(CG_1, CG_2)

begin

$\mathcal{E} = \{\}$; $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$

for each edge $(n_i, n_j) \in \mathcal{E}_1 \cap \mathcal{E}_2$

$\mathcal{E} = \mathcal{E} \cup \{(n_i, n_j)\}$; $w(n_i, n_j) = w_1(n_i, n_j)$

for each edge $(n_i, n_j) \in \mathcal{E}_1 - \mathcal{E}_2$

for each path $(n_i, n_{i+1}, n_{i+2}, \dots, n_j)$ between n_i and n_j in CG_2

for each (n_i, n_{i+1}) in this path

$\mathcal{E} = \mathcal{E} \cup \{(n_i, n_{i+1})\}$; $w(n_i, n_{i+1}) = w_2(n_i, n_{i+1})$

if there is no path between n_i and n_j in CG_2

$\mathcal{E} = \mathcal{E} \cup \{(n_i, n_j)\}$; $w(n_i, n_j) = w_1(n_i, n_j)$

for each edge $(n_i, n_j) \in \mathcal{E}_2 - \mathcal{E}_1$

for each path $(n_i, n_{i+1}, n_{i+2}, \dots, n_j)$ from n_i to n_j in CG_1

for each (n_i, n_{i+1}) in this path

$\mathcal{E} = \mathcal{E} \cup \{(n_i, n_{i+1})\}$; $w(n_i, n_{i+1}) = w_1(n_i, n_{i+1})$

if there is no path from n_i to n_j in \mathcal{CG}_1
 $\mathcal{E} = \mathcal{E} \cup \{(n_i, n_j)\}; w(n_i, n_j) = w_2(n_i, n_j)$

end

6 Evaluating Trust without Complete Information

The model presented so far describes how trust pertaining to some context is evaluated, how trust changes with time, and how different trust vectors can be compared. All this relies on the assumption that the trust vector can be determined in a given context. This may not be possible when complete information is not available. For instance, the trust evaluation policy vector may assign a weight of 0.5 to the recommendation component and it may not be possible to obtain any recommendation about the trustee under the given context. In that case the truster is totally uncertain about the recommendation (recommendation vector is $(0, 0, 1)$ in this case). In the worst case, it may not be possible to obtain information about any of the components for a given context. That is, the truster is totally uncertain about all the parameters. This situation is not very uncommon – suppose a truster is trying to determine the trustworthiness of a new software product. In such a case, the truster may not have any experience or knowledge pertaining to this product. Obtaining a recommendation is also not possible. In such a case, the model that we have presented so far, cannot determine the trust vector of the software product. In this section, we present an approach using which a truster can obtain an approximate trust value of the given software product. Our approach exploits the relationships between contexts in order to extrapolate the values related to trust.

6.1 Extrapolating Trust Values from Related Contexts

When a truster A cannot determine the values related to his trust relationship with trustee B for a context \mathcal{C} , we show how the values can be obtained from one or more related contexts, say, \mathcal{C}_i . The first issue that we must resolve is what values should we use from the related context \mathcal{C}_i . There are two possibilities. We can use the values of the components belief, disbelief, and uncertainty from \mathcal{C}_i to extrapolate the component values for \mathcal{C} . Alternately, we can use the component values of the individual parameters recommendation, experience, and knowledge from \mathcal{C}_i and use these to compute the trust vector for \mathcal{C} . We adopt the second approach for two reasons. First, the trust policy evaluation vector can be different for contexts \mathcal{C} and \mathcal{C}_i .

Using the component values of C_i to extrapolate the component values for C will not be very meaningful. Second, we may not be missing all the components of C . For instance, we may have values for recommendation and knowledge for context C but no value for experience. In such a case, we would want to extrapolate only the experience component from the context C_i .

The second issue is that a context C may be related to many other contexts, say, C_i , C_j , and C_k . Which contexts do we refer to in order to evaluate the trust vector for context C ? Many strategies are possible and different strategies may be needed in different real-world situations. In this paper, we propose a very simple strategy for choosing related contexts. The algorithm given below describes this strategy. This algorithm has three inputs. One is the context c whose closest context we are trying to determine. The other is the context graph CG in which c is a context. The third input is the set of contexts that we should not consider. We term this as the prohibited set of contexts. The algorithm uses a variable *total_weight* that is used for evaluating the closest context. The algorithm proceeds by checking the component contexts of c . If these are not prohibited, then the total weight of all these component contexts makes up the variable *total_weight*. These component contexts are inserted into the set *closest* which contains the closest context. We then check each generalized parent and each specialized children whether the weight on the edge is greater than the *total_weight*. If so, *total_weight* is assigned this new weight and *closest* is initialized with this parent or children. The algorithm returns the set *closest* which gives the set of contexts closest to c .

Algorithm 2 Get the closest context

Input: (i) c – the context whose closest one needs to be determined. (ii) CG – the context graph in which c is a context. (iii) S – set of contexts that should not be considered.

Output: *closest* – set of contexts closest to c

Procedure *ChooseContext*(c, CG, S)

begin

closest = {}; *total_weight* = 0

let CC be the set of component contexts of c

for each $c_i \in CC(c)$

if $c_i \notin S$

total_weight = *total_weight* + $w(c_i, c)$; *closest* = *closest* \cup $\{c_i\}$

for each generalization or composite context p_i of c

if $p_i \notin S$ and *total_weight* < $w(c, p_i)$

```

    total_weight = w(c, p_i); closest = {p_i}
for each specialization r_i of c
    if r_i ∉ S and total_weight < w(r_i, c)
        total_weight = w(r_i, c); closest = {r_i}
    return closest
end

```

Finally, we give an example that shows how recommendation about trustee B can be obtained from the closest context. We describe the steps to be performed in the form of an algorithm.

Algorithm 3 Get Recommendation about Trustee B from the Closest Context

Input: (i) c – the context whose recommendation value needs to be determined. (ii) CG – the context graph in which c is a context.

Output: $(A\psi b_B^c, A\psi d_B^c, A\psi u_B^c)$ – recommendation in context c

Procedure $GetRecommendation(c, CG, S)$

begin

if $(A\psi b_B^c, A\psi d_B^c, A\psi u_B^c) \neq (0, 0, 1)$ **return** $(A\psi b_B^c, A\psi d_B^c, A\psi u_B^c)$

else

$S = \{c\}; closest = \{c\}$

while $(A\psi b_B^c, A\psi d_B^c, A\psi u_B^c) = (0, 0, 1)$ and $closest \neq \{c\}$ **do**

begin

$closest = ChooseClosest(c, CG, S)$

Case 1: $closest = \{c_i\}$ and c is a component or specialization of c_i

if $(A\psi b_B^{c_i}, A\psi d_B^{c_i}, A\psi u_B^{c_i}) \neq (0, 0, 1)$ **then**

$(A\psi b_B^c, A\psi d_B^c, A\psi u_B^c) = w(c, c_i) \times (A\psi b_B^{c_i}, A\psi d_B^{c_i}, A\psi u_B^{c_i})$

$A\psi u_B^c = 1 - A\psi b_B^c - A\psi d_B^c$

return $(A\psi b_B^c, A\psi d_B^c, A\psi u_B^c)$

else $S = S \cup \{c_i\}$

Case 2: $closest = \{c_i\}$ and c is a generalization of c_i

if $(A\psi b_B^{c_i}, A\psi d_B^{c_i}, A\psi u_B^{c_i}) \neq (0, 0, 1)$ **then**

$(A\psi b_B^c, A\psi d_B^c, A\psi u_B^c) = w(c_i, c) \times (A\psi b_B^{c_i}, A\psi d_B^{c_i}, A\psi u_B^{c_i})$

$A\psi u_B^c = 1 - A\psi b_B^c - A\psi d_B^c$

```

    return  $(A_{\Psi}b_B^c, A_{\Psi}d_B^c, A_{\Psi}u_B^c)$ 
else  $S = S \cup \{c_i\}$ 
Case 3:  $closest = \{c_1, c_2, \dots, c_n\}$  and  $c$  is composed of  $c_i$  ( $i = 1, 2, \dots, n$ )
set  $j = 0$  and  $(A_{\Psi}b_B^c, A_{\Psi}d_B^c, A_{\Psi}u_B^c) = (0, 0, 0)$ 
for each  $c_i \in closest$ 
    if  $(A_{\Psi}b_B^{c_i}, A_{\Psi}d_B^{c_i}, A_{\Psi}u_B^{c_i}) \neq (0, 0, 1)$  then
         $(A_{\Psi}b_B^c, A_{\Psi}d_B^c, A_{\Psi}u_B^c) = (A_{\Psi}b_B^{c_i}, A_{\Psi}d_B^{c_i}, A_{\Psi}u_B^{c_i}) + w(c_i, c) \times (A_{\Psi}b_B^{c_i}, A_{\Psi}d_B^{c_i}, A_{\Psi}u_B^{c_i})$ 
         $j = j + 1$ 
    else  $S = S \cup \{c_i\}$ 
     $(A_{\Psi}b_B^c, A_{\Psi}d_B^c, A_{\Psi}u_B^c) = \frac{1}{j} \times (A_{\Psi}b_B^c, A_{\Psi}d_B^c, A_{\Psi}u_B^c)$ 
     $A_{\Psi}u_B^c = 1 - A_{\Psi}b_B^c - A_{\Psi}d_B^c$ 
    return  $(A_{\Psi}b_B^c, A_{\Psi}d_B^c, A_{\Psi}u_B^c)$ 
end
end

```

The above steps show how to extrapolate the values of components of recommendation vector $(A_{\Psi}b_B^c, A_{\Psi}d_B^c, A_{\Psi}u_B^c)$ when it is $(0, 0, 1)$. It does this by getting the recommendation from its closest relation and multiplying it by the weight of the edge. Similar algorithms can be developed for extrapolating the value related to knowledge and experience.

7 Conclusions and Future Work

In this paper we introduce a new model of trust which we term the context-sensitive trust model. Trust is specified as a trust relationship between a truster and a trustee at a particular time instance and for a particular context. We identify three parameters namely, experience, knowledge and recommendation that contribute towards defining this trust relationship. We propose expression for evaluating these factors. Next we introduce the concept of normalized trust. We show how to factor in a notion of trust policy in computing the trust vector. We also model the notion of trust dynamics, that is the change of trust with time. Incorporating all these different notions we finally provide an expression to compute a trust vector that also includes the effect of a previous trust relationship between the same truster, trustee in the same context. We also define ways by which we can compare two trust vectors.

In order to make our trust model interoperable, we formalize the notion of context. A context in our model is described by a set of keywords. Distinct contexts are related using generalization/specialization or composition relationships. Understanding this relationship is important for several reasons. First, it can reason about trust relationships in different contexts, or in cases where the same context is described by different terms. Second, it can extrapolate values related to a trust relationship even when certain information is missing for a given context.

A lot of work remains to be done. We plan to extend this model to define trust combination operators that will allow us to formulate the trust relationship between multiple trusters and trustees. We also plan to formalize the notion of trust chains in the context of our model. In the current work we have not addressed the issue of determining trust policy. We have assumed that there is an underlying trust policy that helps us assign weights to the various components of the model. We plan to identify the factors that influence this policy. We also plan to propose a formal language for managing and manipulating trust relationships. We are looking towards an SQL like language for this purpose. Finally, we plan to develop a complete trust management framework based on our model.

Acknowledgment

This work was partially supported by the U.S. Air Force Research Laboratory (AFRL) and the Federal Aviation Administration (FAA) and the Air Force Office Scientific Research (AFOSR) under contract numbers F30602-03-1-0101 and FA9550-07-1-0042. The views presented here are solely those of the authors and do not necessarily represent those of the AFRL, the AFOSR or the FAA.

References

- [1] A. Abdul-Rahman and S. Hailes. Supporting Trust in Virtual Communities. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pages 1769–1777, Maui, Hawaii, USA, January 2000. IEEE Computer Society.
- [2] M. Bacharach and D. Gambetta. Trust as Type Identification. In C. Castelfranchi and Y. Tan, editors, *Trust and Deception in Virtual Societies*, pages 1–26. Kluwer Academic

- Publishers, 2000.
- [3] T. Beth, M. Borchering, and B. Klein. Valuation of Trust in Open Networks. In Dieter Gollmann, editor, *Proceedings of the 3rd European Symposium on Research in Computer Security*, volume 875 of *Lecture Notes in Computer Science*, pages 3–18, Brighton, United Kingdom, November 1994. Springer-Verlag.
 - [4] M. Burrows, M. Abadi, and R.M. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.
 - [5] M.S. Cohen, R. Parasuraman, R.S. Serfaty, and R.C. Andes. Trust in Decision Aids: A Model and a Training Strategy. Technical Report USAATCOM TR 97-D-4, Cognitive Technologies Inc., Fort Eustis, Virginia, USA, 1997.
 - [6] T. Grandison and M. Sloman. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4):2–16, Fourth Quarter 2000.
 - [7] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
 - [8] S. Jajodia, P. Samarati, and V. Subrahmanian. A Logical Language for Expressing Authorizations. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 31–42, Oakland, California, USA, May 1997. IEEE Computer Society.
 - [9] A.J.I. Jones and B.S. Firozabadi. On the Characterization of a Trusting Agent – Aspects of a Formal Approach. In C.Castelfranchi and Y.Tan, editors, *Trust and Deception in Virtual Societies*, pages 163–174. Kluwer Academic Publishers, 2000.
 - [10] A. Jøsang. Artificial Reasoning with Subjective Logic. In *Proceedings of the Second Australian Workshop on Commonsense Reasoning*, Perth, Australia, December 1997.
 - [11] A. Jøsang. A Subjective Metric of Authentication. In Jean-Jacques Quisquater et al., editors, *Proceedings of the 5th European Symposium on Research in Computer Security*, volume 1485 of *Lecture Notes in Computer Science*, pages 329–344, Louvain-la-Neuve, Belgium, September 1998. Springer-Verlag.
 - [12] A. Jøsang. An Algebra for Assessing Trust in Certification Chains. In *Proceedings of Network and Distributed Systems Security Symposium*, San Diego, California, USA, February 1999. Internet Society.
 - [13] A. Jøsang. A Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, June 2001.

- [14] A. Jøsang, E. Gray, and M. Kinaterer. Simplification and Analysis of Transitive Trust Networks. *Web Intelligence and Agent Systems Journal*, 4(2):139–161, 2006.
- [15] Li Xiong Li and Ling Liu. A Reputation-Based Trust Model For Peer-To-Peer Ecommerce Communities. In *Proceedings of IEEE Conference on E-Commerce*, pages 275–284, Newport Beach, California, USA, June 2003. IEEE Computer Society.
- [16] S. Purser. A Simple Graphical Tool For Modelling Trust. *Computers & Security*, 20(6):479–484, September 2001.
- [17] P.V. Rangan. An Axiomatic Basis of Trust in Distributed Systems. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pages 204–211, Oakland, California, USA, April 1988. IEEE Computer Society.
- [18] I. Ray and S. Chakraborty. A Vector Model of Trust for Developing Trustworthy Systems. In P. Samarati, P. Ryan, D. Gollmann, and R. Molva, editors, *Proceedings of the 9th European Symposium on Research in Computer Security*, volume 3193 of *Lecture Notes In Computer Science*, pages 260–275, Sophia Antipolis, Frech Riviera, France, September 2004. Springer-Verlag.
- [19] Indrajit Ray, Sudip Chakraborty, and Indrakshi Ray. VTrust: A Trust Management System Based on a Vector Model of Trust. In Sushil Jajodia and Chandan Mazumdar, editors, *Proceedings of 1st International Conference on Information Systems Security*, volume 3803 of *Lecture Notes in Computer Science*, pages 91–105, Kolkata, India, December 2005. Springer-Verlag GmbH.
- [20] Michael Stevens and Paul D. Williams. Use of Trust Vectors for CyberCraft and the Limits of Usable Data History for Trust Vectors. In *2007 IEEE Computational Intelligence for Security and Defense Applications*, Honolulu, Hawaii, USA, April 2007.
- [21] M. Uschold and M. Grüninger. Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.
- [22] R. Yahalom and B. Klein. Trust-based Navigation in Distributed Systems. *Computing Systems*, 7(1):45–73, Winter 1994.
- [23] R. Yahalom, B. Klein, and T. Beth. Trust Relationship in Secure Systems: A Distributed Authentication Perspective. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 150–164, Oakland, California, USA, May 1993. IEEE Computer Society.