# Trust Network Analysis with Subjective Logic

**Audun Jøsang**[1]        **Ross Hayward**[1]        **Simon Pope**[2]

[1]School of Software Engineering and Data Communications[*]
Queensland University of Technology, Brisbane, Australia
Email: {a.josang, r.hayward}@qut.edu.au

[2]CRC for Enterprise Distributed Systems Technology (DSTC Pty Ltd)[†]
The University of Queensland, Brisbane, Australia
Email: skjpope@gmail.com

## Abstract

Trust networks consist of transitive trust relationships between people, organisations and software agents connected through a medium for communication and interaction. By formalising trust relationships, e.g. as reputation scores or as subjective trust measures, trust between parties within the community can be derived by analysing the trust paths linking the parties together. This article describes a method for trust network analysis using subjective logic (TNA-SL). It provides a simple notation for expressing transitive trust relationships, and defines a method for simplifying complex trust networks so that they can be expressed in a concise form and be computationally analysed. Trust measures are expressed as beliefs, and subjective logic is used to compute trust between arbitrary parties in the network. We show that TNA-SL is efficient, and illustrate possible applications with examples.

## 1 Introduction

Modern communication media are increasingly removing us from the familiar styles of interacting that traditionally rely on some degree of pre-established trust between business partners. Moreover, most traditional cues for assessing trust in the physical world are not available through those media. We may now be conducting business with people and organisations of which we know nothing, and we are faced with the difficult task of making decisions involving risk in such situations. As a result, the topic of trust in open computer networks is receiving considerable attention in the network security community and e-commerce industry [1, 4, 13, 18, 19, 23, 26]. State of the art technology for stimulating trust in e-commerce includes cryptographic security mechanisms for providing confidentiality of communication and authentication of identities. However, merely having a cryptographically certified identity or knowing that the communication channel is encrypted is not enough for making informed decisions if no other knowledge about a remote transaction partner is available. Trust therefore also applies to the truthfulness of specific claims made by parties who request services in a given business context as described in

the WS-Trust specifications [26], and trust between business partners regarding security assertions as described in the Liberty Alliance Framework [18, 19]. Trust also applies to the honesty, reputation and reliability of service providers or transaction partners, in general or for a specific purpose. In this context, the process of assessing trust becomes part of quality of service (QoS) evaluation, decision making and risk analysis.

Being able to formally express and reason with these types of trust is needed not only to create substitutes for the methods we use in the physical world, like for instance trust based on experiences or trust in roles, but also for creating entirely new methods for determining trust that are better suited for computerised interactions. This will facilitate the creation of communication infrastructures where trust can thrive in order to ensure meaningful and mutually beneficial interactions between players.

The main contribution of this paper is a method for discovering trust networks between specific parties, and a practical method for deriving measures of trust from such networks. Our method, which is called TNA-SL (Trust Network Analysis with Subjective Logic), is based on analysing trust networks as directed series-parallel graphs that can be represented as canonical expressions, combined with measuring and computing trust using subjective logic. We finally provide a numerical example of how trust can be analysed and computed using our method.

## 2 Trust Transitivity

Trust transitivity means, for example, that if Alice trusts Bob who trusts Eric, then Alice will also trust Eric. This assumes that Bob actually tells Alice that he trusts Eric, which is called a *recommendation*.

It can be shown that trust is not always transitive in real life [2]. For example the fact that Alice trusts Bob to look after her child, and Bob trusts Eric to fix his car, does not imply that Alice trusts Eric for looking after her child, or for fixing her car. However, under certain semantic constraints [15], trust can be transitive, and a trust system can be used to derive trust. In the last example, trust transitivity collapses because the scopes of Alice's and Bob's trust are different.

We define *trust scope*[1] as the specific type(s) of trust assumed in a given trust relationship. In other words, the trusted party is relied upon to have certain qualities, and the scope is what the trusting party assumes those qualities to be.

Let us assume that Alice needs to have her car serviced, so she asks Bob for his advice about where to find a good car mechanic in town. Bob is thus trusted by Alice to know about a good car mechanic and to tell his honest

---

[1]The terms "trust context" [6], "trust purpose" [13] and "subject matter" [20] have been used in the literature with the same meaning.
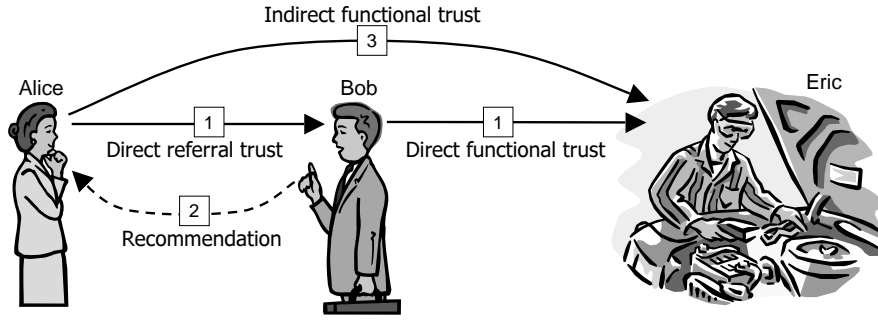
Figure 1: Transitive trust principle

opinion about that. Bob in turn trusts Eric to be a good car mechanic. This situation is illustrated in Fig.1, where the indexes indicate the order in which the trust relationships and recommendations are formed.

It is important to separate between trust in the ability to recommend a good car mechanic which represents *referral trust*, and trust in actually being a good car mechanic which represents *functional trust*. The scope of the trust is nevertheless the same, namely to be a good car mechanic. Assuming that, on several occasions, Bob has proved to Alice that he is knowledgeable in matters relating to car maintenance, Alice's referral trust in Bob for the purpose of recommending a good car mechanic can be considered to be *direct*. Assuming that Eric on several occasions has proved to Bob that he is a good mechanic, Bob's functional trust in Eric can also be considered to be direct. Thanks to Bob's advice, Alice also trusts Eric to actually be a good mechanic. However, this functional trust must be considered to be *indirect*, because Alice has not directly observed or experienced Eric's skills in car mechanics.

Let us slightly extend the example, wherein Bob does not actually know any car mechanics himself, but he knows Claire, whom he believes knows a good car mechanic. As it happens, Claire is happy to recommend the car mechanic named Eric. As a result of transitivity, Alice is able to derive trust in Eric, as illustrated in Fig.2, where the indexes indicate the order in which the trust relationships and recommendations are formed. The prefix "dr-" denotes direct referral trust, "df-" denotes direct functional trust, and "if-" denotes indirect functional trust.
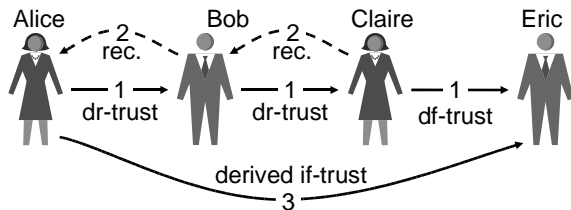


Figure 2: Transitive serial combination of trust arcs

Defining the exact scope of Alice's trust in Bob is more complicated in the extended example. It is most obvious to say that Alice trusts Bob to recommend somebody (who can recommend somebody etc.) who can recommend a good car mechanic. The problem with this type of formulation is that the length of the trust scope expression becomes proportional with the length of the transitive path, so that the trust scope expression rapidly becomes impenetrable. It can be observed that this type of trust scope has a recursive structure that can be exploited to define a more compact expression for the trust scope. As already mentioned, trust in the ability to recommend represents referral trust, and is precisely what allows trust to become transitive. At the same time, referral trust always assumes the existence of a functional trust scope at the end of the tran-

sitive path, which in this example is about being a good car mechanic.

The "referral" variant of a trust scope can be considered to be recursive, so that any transitive trust chain, with arbitrary length, can be expressed using only one trust scope with two variants. This principle is captured by the following criterion.

**Definition 1 (Functional Trust Derivation Criterion)**
*Derivation of functional trust through referral trust, requires that the last trust arc represents functional trust, and all previous trust arcs represent referral trust.*

In practical situations, a trust scope can be characterised by being general or specific. For example, knowing how to change wheels on a car is more specific than to be a good car mechanic, where the former scope is a subset of the latter. Whenever a given trust scope is part of all the referral and functional trust scopes in a path, a transitive trust path can be formed based on that trust scope. This can be expressed with the following consistency criterion.

**Definition 2 (Trust Scope Consistency Criterion)** *A valid transitive trust path requires that there exists a trust scope which is a common subset of all trust scopes in the path. The derived trust scope is then the largest common subset.*

Trivially, every arc in a path can carry the same trust scope. Transitive trust propagation is thus possible with two variants (i.e. functional and referral) of a single trust scope.

Specifying the two scope variants separately can be omitted in case it is difficult to separate between them in a given application. Although trust scopes are always expressed with the two variants in all the descriptions and example of this paper, it is perfectly possible to assume the same descriptions and examples without specifying the two variants.

A transitive trust path stops with the first functional trust arc encountered when there are no remaining outgoing referral trust arcs. It is, of course, possible for a principal to have both functional and referral trust in another principal, but that should be expressed as two separate trust arcs.

The examples above assume some sort of absolute trust between the agents along the transitive trust path. In reality trust is never absolute, and many researchers have proposed to express trust as discrete verbal statements, as probabilities or other continuous measures. One observation which can be made from an intuitive perspective is that trust is diluted through transitivity. Revisiting the example of Fig.2, it can be noted that Alice's trust in the car

mechanic Eric through the recommenders Bob and Claire can be at most as confident as Claire's trust in Eric.

It could be argued that negative trust in a transitive chain can have the paradoxical effect of strengthening the derived trust. Take for example the case where Bob distrusts Claire and Claire distrusts Eric, whereas Alice trusts Bob. In this situation, Alice might actually derive positive trust in Eric, since she relies on Bob's advice, and Bob says: *"Claire is a cheater, do not rely on her"*. So the fact that Claire distrusts Eric might count as a pro-Eric argument from Alice's perspective. The question boils down to *"is the enemy of my enemy my friend?"*. However this question relates to how multiple types of untrustworthiness, such as dishonesty and unreliability, should be interpreted in a trust network, which is outside the scope of this study.

## 3   Parallel Trust Combination

It is common to collect advice from several sources in order to be better informed when making decisions. This can be modelled as *parallel trust combination* illustrated in Fig.3, where again the indexes indicate the order in which the trust relationships and recommendations are formed.
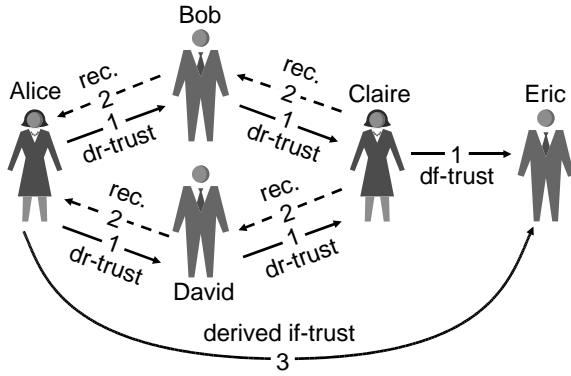


Figure 3: Parallel combination of trust paths

Let us assume again that Alice needs to get her car serviced, and that she asks Bob to recommend a good car mechanic. When Bob replies that Claire, a good friend of his, recommended Eric to him, Alice would like to get a second opinion, so she asks David whether he has heard about Eric. David also knows and trusts Claire, and has heard from her that Eric is a good car mechanic. Alice who does not know Claire personally, is unable to obtain a first hand recommendation about the car mechanic Eric, i.e. she does not directly know anybody with functional trust in Eric. Intuitively, if both Bob and David recommend Claire as a good advisor regarding car mechanics, Alice's trust in Claire's advice will be stronger than if she had only asked Bob. Parallel combination of positive trust thus has the effect of strengthening the derived trust.

In the case where Alice receives conflicting recommended trust, e.g. trust and distrust at the same time, she needs some method for combining these conflicting recommendations in order to derive her trust in Eric. Our method, which is described in Sec.6, is based on subjective logic which easily can handle such cases.

## 4   Structured Notation

Transitive trust networks can involve many principals, and in the examples below, capital letters $A, B, C, D$ and $E$ will be used to denote principals instead of names such as Alice and Bob.

We will use basic constructs of directed graphs to represent transitive trust networks. We will add some notation elements which allow us to express trust networks in a structured way.

A single trust relationship can be expressed as a directed arc between two nodes that represent the trust source and the trust target of that arc. For example the arc $[A, B]$ means that $A$ trusts $B$.

The symbol ":" will be used to denote the transitive connection of two consecutive trust arcs to form a transitive trust path. The trust relationships of Fig.2 can be expressed as:

$$([A, E]) = ([A, B] : [B, C] : [C, E]) \qquad (1)$$

where the trust scope is implicit. Let the trust scope e.g. be defined as $\sigma$: *"trust to be a good car mechanic"*. Let the functional variant be denoted by "f$\sigma$" and the referral variant by "r$\sigma$". A distinction can be made between initial *direct trust* and derived *indirect trust*. Whenever relevant, the trust scope can be prefixed with "d" to indicate direct trust (d$\sigma$), and with "i" to indicate indirect trust (i$\sigma$). This can be combined with referral and functional trust, so that for example indirect functional trust can be denoted as "if$\sigma$". A reference to the trust scope can then be explicitly included in the trust arc notation as e.g. denoted by $[A, B, \mathrm{dr}\sigma]$. The trust network of Fig.2 can then be explicitly expressed as:

$$([A, E, \mathrm{if}\sigma]) = \\ ([A, B, \mathrm{dr}\sigma] : [B, C, \mathrm{dr}\sigma] : [C, E, \mathrm{df}\sigma]) \qquad (2)$$

Let us now turn to the combination of parallel trust paths, as illustrated in Fig.3. We will use the symbol "$\diamond$" to denote the graph connector for this purpose. The "$\diamond$" symbol visually resembles a simple graph of two parallel paths between a pair of agents, so that it is natural to use it for this purpose. Alice's combination of the two parallel trust paths from her to Eric in Fig.3 is then expressed as:

$$([A, E, \mathrm{if}\sigma]) = \quad ((([A, B, \mathrm{dr}\sigma] : [B, C, \mathrm{dr}\sigma]) \diamond \\ ([A, D, \mathrm{dr}\sigma] : [D, C, \mathrm{dr}\sigma])) : \quad (3) \\ [C, E, \mathrm{df}\sigma])$$

In short notation, the same trust graph is expressed as:

$$([A, E]) = \quad ((([A, B] : [B, C]) \diamond \\ ([A, D] : [D, C])) : [C, E]) \qquad (4)$$

It can be noted that Fig.3 contains two paths. The graph consisting of the two separately expressed paths would be:

$$([A, E]) = \quad ([A, B] : [B, C] : [C, E]) \diamond \\ ([A, D] : [D, C] : [C, E]) \qquad (5)$$

A problem with Eq.(5) is that the arc $[C, E]$ appears twice. Although Eq.(4) and Eq.(5) consist of the same two paths, their combined structures are different. Some computational models would be indifferent to Eq.(4) and Eq.(5), whereas others would produce different results depending on which expression is being used. When implementing the serial ":" as binary logic "AND", and the parallel "$\diamond$" as binary logic "OR", the results would be equal. However, when implementing ":" and "$\diamond$" as probabilistic multiplication and comultiplication respectively, the results would be different. It would also be different in

the case of applying subjective logic operators for transitivity and parallel combination which will be described in Sec.6 below. In general, it is therefore desirable to express graphs in a form where an arc only appears once. This will be called a *canonical expression*.

**Definition 3 (Canonical Expression)** *An expression of a trust graph in structured notation where every arc only appears once is called canonical.*

With this structured notation, arbitrarily large trust networks can be explicitly expressed in terms of source, target, and scope, as well as other attributes such as measure and time whenever required.

A general directed trust graph is based on directed trust arcs between pairs of nodes. With no restrictions on the possible trust arcs, trust paths from a given source $X$ to a given target $Y$ can contain cycles, which could result in inconsistent calculative results. Cycles in the trust graph must therefore be controlled when applying calculative methods to derive measures of trust between two parties. *Normalisation* and *simplification* are two different control approaches. Our model is based on graph simplification, and a comparison with normalisation methods used in e.g. PageRank proposed by Page *et al.* (1998) [21], and in EigenTrust proposed by Kamvar *et al.* (2003) [17] is provided in [10].

## 5 Network Simplification

Simplification of a trust network consists of including as many arcs as possible from the original trust network, while still maintaining a canonical expression. Graphs that can be represented as canonical expressions with our structured notation are known as *directed series-parallel graphs* (DSPG) [5]. A DSPG can be constructed by sequences of serial and parallel compositions that are defined as follows [5]:

**Definition 4 (Directed Series-Parallel Composition)**

- *A directed series composition consists of replacing an arc $[A, C]$ with two arcs $[A, B]$ and $[B, C]$ where $B$ is a new node.*

- *A directed parallel composition consists of replacing an arc $[A, C]$ with two arcs $[A, C]_1$ and $[A, C]_2$.*

The principle of directed series and parallel composition are illustrated in Fig.4.



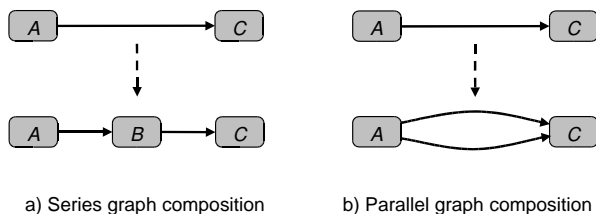a) Series graph composition    b) Parallel graph composition

Figure 4: DSPG composition.

By successively applying the principles of series and parallel composition, arbitrarily large DSPGs can be constructed.

We will first describe an algorithm for determining all practical trust paths from a given source to a given target, and secondly algorithms for determining near-optimal or optimal DSPGs.

## 5.1 Finding Paths

The first step is to determine the possible directed paths between a given pair of agents called the start source and the final target. The pseudo-code in Fig.5 represents an algorithm for finding all practical directed paths between a given start source and a given final target, where no single path contains cycles.

```
Pseudo-Constructor for a trust arc between two parties:

Arc(Node source, Node target, Scope scope, Variant variant){
    this.source = source;
    this.target = target;
    this.scope = scope;
    this.variant = variant;
}


Pseudo-code for a depth-first path finding algorithm:
After completion, 'paths' contains all possible paths between source
and target.

void FindPaths(Node source, Node target, Scope scope) {
    SELECT arcs FROM graph WHERE (
        (arcs.source == source) AND
        (arcs.target NOT IN path) AND
        (arcs.scope == scope))
    FOR EACH arc IN arcs DO {
        IF (
            (arc.target == target) AND
            (arc.variant == 'functional') AND
            (Confidence(path + arc) > Threshold)) {
            paths.add(path + arc);
        }
        ELSE IF (
            (arc.target != target) AND
            (arc.variant == 'referral') AND
            (Confidence(path + arc) > Threshold)) {
        path.add(arc);
        FindPaths(arc.target, target, scope);
        path.remove(arc);
        }
    }
}


Pseudo-code for method call:
The global variables 'path' and 'paths' are initialized.

Vector path = NEW Vector OF TYPE arc;
Vector paths = NEW Vector OF TYPE path;
FindPaths(StartSource, FinalTarget, scope);
```

Figure 5: Path finding algorithm

In the pseudocode of Fig.5, the conditional

$$\text{IF (Confidence(path + arc)} > \text{Threshold)}$$

represents a heuristic rule for simplifying the graph analysis, where the path is only retained as long as the conditional is TRUE. By removing paths with low confidence, the number of paths to consider is reduced while the information loss can be kept to an insignificant level. For a given application, the threshold can be defined as the lowest level for which a trust relationship is meaningful. The mathematical interpretation of confidence is described in Sec.6.1.

## 5.2 Finding Directed Series-Parallel Graphs

Ideally, all the possible paths discovered by the algorithm of Fig.5 should be taken into account when deriving the trust value. A general directed graph will often contain cycles and dependencies. This can be avoided by excluding certain paths, but this can also cause information loss.

Specific selection criteria are needed in order to find the optimal subset of paths to include.

Fig.6 illustrates an example of a non-DSPG with dependent paths, where it is assumed that $A$ is the source and $E$ is the target. While there can be a large number of possible distinct paths, it is possible to use heuristic rules to discard paths, e.g. when their confidence drops below a certain threshold.
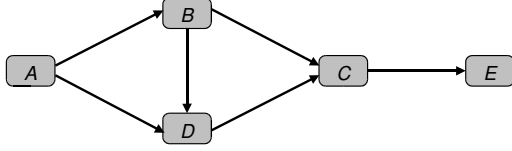


Figure 6: Dependent paths

With $n$ possible paths, there are $2^n - 1$ different combinations for constructing graphs, of which not all necessarily are DSPGs. Of the graphs that are DSPGs, only one will be selected for deriving the trust measure.

In Fig.6 there are 3 possible paths between $A$ and $E$:

$$\begin{aligned}
\phi_1 &= ([A,B]:[B,C]:[C,E]), \\
\phi_2 &= ([A,D]:[D,C]:[C,E]), \\
\phi_3 &= ([A,B]:[B,D]:[D,C]:[C,E]).
\end{aligned} \quad (6)$$

This leads to the following 7 potential combinations/graphs.

$$\begin{aligned}
&\gamma_1 = \phi_1, \quad &\gamma_4 = \phi_1 \diamond \phi_2, \quad &\gamma_7 = \phi_1 \diamond \phi_2 \diamond \phi_3. \\
&\gamma_2 = \phi_2, \quad &\gamma_5 = \phi_1 \diamond \phi_3, \\
&\gamma_3 = \phi_3, \quad &\gamma_6 = \phi_2 \diamond \phi_3,
\end{aligned} \quad (7)$$

The graph represented by $\gamma_7$ contains all possible paths between $A$ and $E$. The problem with $\gamma_7$ is that it can not be represented as a canonical expression, i.e. where an arc can only appear once. In this example, one path must must be removed from the graph in order to have a canonical expression. The expressions $\gamma_4$, $\gamma_5$ and $\gamma_6$ can be canonicalised, and the expressions $\gamma_1$, $\gamma_2$ and $\gamma_3$ are already canonical, which means that all the expressions except $\gamma_7$ can be used as a basis for constructing a DSPG and for deriving $A$'s trust in $E$.

The optimal DSPG is the one that results in the highest confidence level of the derived trust value. This principle focuses on maximising certainty in the trust value, and not e.g. on deriving the most positive or negative trust value. The interpretation of confidence can of course have different meanings depending on the computational model, and our approach is based on he classic confidence value of probability density functions.

There is a trade-off between the time it takes to find the optimal DSPG, and how close to the optimal DSPG a simplified graph can be. It is possible to use a relatively fast *heuristic algorithm* to find a DSPG close to, or equal to the optimal DSPG. It is also possible to use a relatively slow *exhaustive algorithm* that is guaranteed to find the optimal DSPG.

### 5.2.1 Heuristic Search for Near-Optimal DSPGs

Fig.7 represents a heuristic algorithm for finding a near-optimal DSPG. It constructs the DSPG by including new paths one by one in decreasing order of confidence. Each new path that potentially could turn the graph into a non-DSPG and break canonicity is excluded. This is detected by analysing each new potential branch with the method:

dspg.sep_subgraph(branch.source,branch.sink)

```
Pseudo-code search algorithm for a near optimal DSPG:
After completion, 'dspg' contains a near-optimal trust graph

void FindNearOptimalDSPG(Vector paths) {
    paths.sort_according_to_confidence;
    dspg = paths(0);
    paths.remove(0);
    FOR EACH path IN paths DO {
        end_of_path = FALSE;
        branch = EMPTY;
        WHILE NOT end_of_path DO {
            next_arc = path.next;
            end_of_path = path.no_more_arcs;
            IF (next_arc.sink NOT IN dspg) {
                branch.add(next_arc);
            }
            ELSE IF ((next_arc.sink IN dspg) AND
                    (branch != EMPTY)) {
                branch.add(next_arc);
                IF (dspg.sep_subgraph(branch.source,branch.sink) {
                    dspg.add(branch);
                    branch = EMPTY;
                }
                ELSE {
                    end_of_path = TRUE;
                }
            }
        }
    }
}

Pseudo-code for method call:
The global variables 'dspg' and 'paths' are initialized.

Vector dspg = NEW Vector OF TYPE arc;
Vector paths = NEW Vector OF TYPE path;
FindNearOptimalDSPG(paths);
```

Figure 7: Heuristic algorithm for a near-optimal DSPG

which returns TRUE if the new branch can be added, and FALSE if not. More precisely, it verifies that the subgraph between the nodes where the new branch is to be added is a separate sub-DSPG, so that a clean parallel graph composition according to Fig.4 is possible when adding the new branch. While this subgraph analysis can be computationally intensive, efficiency can be improved by caching these intermediate results, so that in case several new branches between the same nodes must be added, the analysis of the corresponding subgraph only needs to be done once.

This method only requires the computation of the trust value for a single DSPG, with computational complexity $\text{Comp} = lm$, where $m$ is average number of paths in the DSPGs, and $l$ is the average number of arcs in the paths.

The heuristic method produces a DSPG with overall confidence in the trust level equal or close to that of the optimal DSPG. The reason why this method can not guarantee to produce the optimal DSPG, is that it could exclude two or more paths with relatively low confidence levels because of conflict with a single path with high confidence level previously included, whereas the low confidence paths together could provide higher confidence than the previous high confidence path alone. In such cases it would have been optimal to exclude the single high confidence path, and instead include the low confidence paths. However, only the exhaustive method described below can guarantee to find the optimal DSPG in such cases.

### 5.2.2 Exhaustive Search for the Optimal DSPG

The exhaustive method of finding the optimal DSPG consists of determining all possible DSPGs, then deriving the trust value for each one of them, and finally selecting the

DSPG and the corresponding canonical expression that produces the trust value with the highest confidence level.

For brevity, we have not included the pseudocode algorithm for the exhaustive search algorithm, because it would be similar to the heuristic search algorithm. The main difference is that all $2^n - 1$ possible orders of including the paths are tried one by one, potentially leading to $2^n - 1$ different DSPGs that must be evaluated. Normally, the DSPG that produces the highest confidence is finally selected.

The computational complexity of the exhaustive method is $\text{Comp} = lm(2^n - 1)$, where $n$ is the number of possible paths, $m$ is the average number of paths in the DSPGs, and $l$ is the average number of arcs in the paths.

## 6  Trust Derivation with Subjective Logic

Subjective logic represents a practical belief calculus that can be used for calculative analysis trust networks. TNA-SL requires trust relationships to be expressed as beliefs, and trust networks to be expressed as DSPGs in the form of canonical expressions. In this section we describe how trust can be derived with the belief calculus of subjective logic. A numerical example is given in Sec.7.

### 6.1  Subjective Logic Fundamentals

Belief theory is a framework related to probability theory, but where the probabilities over the set of possible outcomes do not necessarily add up to 1, and the remaining probability is assigned to the union of possible outcomes. Belief calculus is suitable for approximate reasoning in situations of partial ignorance regarding the truth of a given proposition.

Subjective logic [7] represents a specific belief calculus that uses a belief metric called *opinion* to express beliefs. An opinion denoted by $\omega_x^A = (b, d, u, a)$ expresses the relying party $A$'s belief in the truth of statement $x$. When a statement for example says *"Party X is honest and reliable regarding $\sigma$"*, then the opinion about the truth of that statement can be interpreted as trust in $X$ within the scope of $\sigma$. Here $b$, $d$, and $u$ represent belief, disbelief and uncertainty respectively, where $b, d, u \in [0, 1]$ and $b + d + u = 1$. The confidence parameter used in the pseudocode of Fig.fig:find-path can be defined as equal to $(1 - c)$, i.e. the confidence of a trust value is equivalent to the certainty of the corresponding opinion. The parameter $a \in [0, 1]$ is called the base rate, and is used for computing an opinion's probability expectation value that can be determined as $\text{E}(\omega_x^A) = b + au$. More precisely, $a$ determines how uncertainty shall contribute to the probability expectation value $\text{E}(\omega_x^A)$. In the absence of any specific evidence about a given party, the base rate determines the *a priori* trust that would be put in any member of the community.

The opinion space can be mapped into the interior of an equal-sided triangle, where, for an opinion $\omega_x = (b_x, d_x, u_x, a_x)$, the three parameters $b_x$, $d_x$ and $u_x$ determine the position of the point in the triangle representing the opinion. Fig.8 illustrates an example where the opinion about a proposition $x$ from a binary state space has the value $\omega_x = (0.7, 0.1, 0.2, 0.5)$.

The top vertex of the triangle represents uncertainty, the bottom left vertex represents disbelief, and the bottom right vertex represents belief. The parameter $b_x$ is the value of a linear function on the triangle which takes value 0 on the edge which joins the uncertainty and disbelief vertexes and takes value 1 at the belief vertex. In other words, $b_x$ is equal to the quotient when the perpendicular distance between the opinion point and the edge joining the uncertainty and disbelief vertexes is divided by
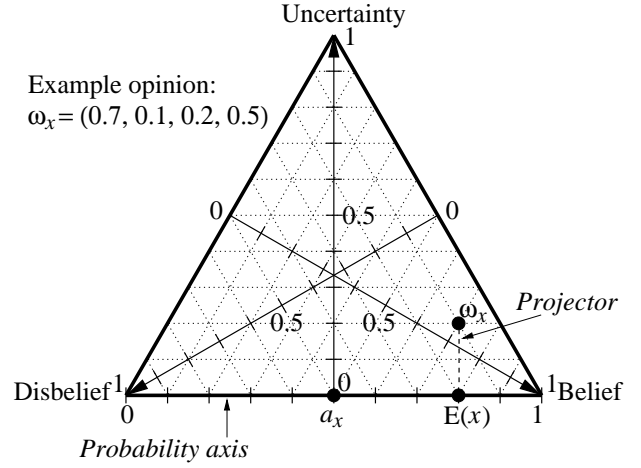


Figure 8: Opinion triangle with example opinion

the perpendicular distance between the belief vertex and the same edge. The parameters $d_x$ and $u_x$ are determined similarly. The base of the triangle is called the probability axis. The base rate is indicated by a point on the probability axis, and the projector starting from the opinion point is parallel to the line that joins the uncertainty vertex and the base rate point on the probability axis. The point at which the projector meets the probability axis determines the expectation value of the opinion, i.e. it coincides with the point corresponding to expectation value $\text{E}(\omega_x^A)$.

Opinions can be ordered according to probability expectation value, but additional criteria are needed in case of equal probability expectation values. We will use the following rules to determine the order of opinions [7]:

Let $\omega_x$ and $\omega_y$ be two opinions. They can be ordered according to the following rules by priority:

1. The opinion with the greatest probability expectation is the greatest opinion.

2. The opinion with the least uncertainty is the greatest opinion.

3. The opinion with the least base rate is the greatest opinion.

The probability density over binary event spaces can be expressed as beta PDFs (probability density functions) denoted by beta $(\alpha, \beta)$ [3]. Let $r$ and $s$ express the number of positive and negative past observations respectively, and let $a$ express the *a priori* or base rate, then $\alpha$ and $\beta$ can be determined as:

$$\alpha = r + 2a , \qquad \beta = s + 2(1 - a) . \tag{8}$$

The following bijective mapping between the opinion parameters and the beta PDF parameters can be determined analytically [7, 14].

$$\left\{ \begin{array}{l} b_x = r/(r + s + 2) \\ d_x = s/(r + s + 2) \\ u_x = 2/(r + s + 2) \\ a_x = \text{base rate of } x \end{array} \right. \Longleftrightarrow \left\{ \begin{array}{l} r = 2b_x/u_x \\ s = 2d_x/u_x \\ 1 = b_x + d_x + u_x \\ a = \text{base rate of } x \end{array} \right. \tag{9}$$

This means for example that a totally ignorant opinion with $u_x = 1$ and $a_x = 0.5$ is equivalent to the uniform PDF beta $(1, 1)$ illustrated in Fig.9.

It also means that a dogmatic opinion with $u_x = 0$ is equivalent to a spike PDF with infinitesimal width and infinite height expressed by beta $(b_x\eta, d_x\eta)$, where $\eta \rightarrow \infty$. Dogmatic opinions can thus be interpreted as being based on an infinite amount of evidence.
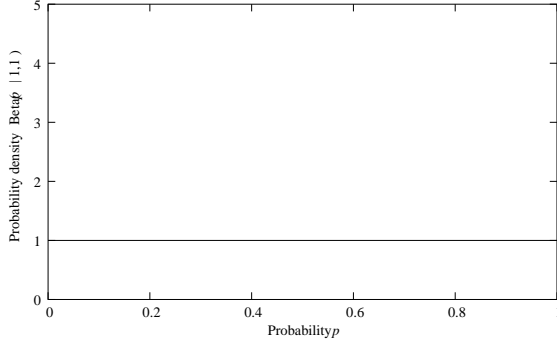
Figure 9: *A priori* uniform beta(1,1)

After $r$ positive and $s$ negative observations in case of a binary state space (i.e. $a = 0.5$), the *a posteriori* distribution is the beta PDF with $\alpha = r + 1$ and $\beta = s + 1$. For example the beta PDF after observing 7 positive and 1 negative outcomes is illustrated in Fig.10, which also is equivalent to the opinion illustrated in Fig.8
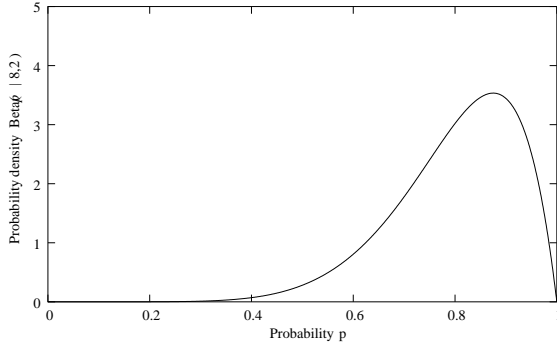


Figure 10: *A posteriori* beta(8,2) after 7 positive and 1 negative observations

A PDF of this type expresses the uncertain probability that a process will produce positive outcome during future observations. The probability expectation value of Fig.10 is $\mathrm{E}(p) = 0.8$. This can be interpreted as saying that the relative frequency of a positive outcome in the future is somewhat uncertain, and that the average value is 0.8.

The variable $p$ is a probability variable, so that for a given $p$ the probability density beta$(\alpha, \beta)$ represents second order probability. The first-order variable $p$ represents the probability of an event, whereas the density beta$(\alpha, \beta)$ represents the probability that the first-order variable has a specific value. Since the first-order variable $p$ is continuous, the second-order probability beta$(\alpha, \beta)$ for any given value of $p \in [0, 1]$ is vanishingly small and therefore meaningless as such. It is only meaningful to compute $\int_{p_1}^{p_2}$ beta$(\alpha, \beta)$ for a given interval $[p_1, p_2]$, or simply to compute the expectation value of $p$. The expectation value of the PDF is always equal to the expectation value of the corresponding opinion. This provides a sound mathematical basis for combining opinions using Bayesian updating of beta PDFs.

## 6.2 Determining Trust with Reputation Systems

The trust representation of subjective logic is directly compatible with the reputation representation of Bayesian reputation systems [12, 13, 25, 24]. This makes it possible to use reputation systems to determine trust measures. The method for doing this is briefly described below.

Bayesian reputation systems allow agents to rate other agents, both positively and negatively, by arbitrary amounts, for a single transaction. This rating takes the form of a vector:

$$\rho = \left[ \begin{array}{c} r \\ s \end{array} \right], \text{ where } r \geq 0 \text{ and } s \geq 0. \tag{10}$$

A simple binary rating system can e.g. be implemented by using $\rho^+ = [1, 0]$ for a satisfactory transaction and $\rho^- = [0, 1]$ for an unsatisfactory transaction [11].

A particular rating can be denoted as:

$$\rho^X_{Z,t_R} \tag{11}$$

which can be read as $X$'s rating of $Z$ at time $t_R$. Whenever not relevant, these super- and subscripts can be omitted.

### 6.2.1 Aging Ratings

Agents (and in particular human agents) may change their behaviour over time, so it is desirable to give greater weight to more recent ratings. This can be achieved by introducing a longevity factor $\lambda$, which controls the rate at which old ratings are 'forgotten':

$$\rho^{X,t}_{Z,t_R} = \lambda^{t-t_R} \rho^X_{Z,t_R} \tag{12}$$

where $0 \leq \lambda \leq 1$, $t_R$ is the time at which the rating was collected and $t$ is the current time.

### 6.2.2 Aggregating Ratings

Ratings may be aggregated by simple addition of the components (vector addition).

For each pair of agents $(X, Z)$, an aggregate rating $\rho^t(X, Z)$ can be calculated that reflects X's overall opinion of Z at time $t$:

$$\rho^t(X, Z) = \sum \rho^{X,t}_{Z,t_R}, \text{ where } t_R \leq t . \tag{13}$$

Also, $Z$'s aggregate rating by all agents in a particular set $S$ can be calculated:

$$\rho^t(Z) = \sum_{X \in S} \rho^t(X, Z). \tag{14}$$

In particular, the aggregate rating for Z, taking into account ratings by the entire agent community $C$, can be calculated:

$$\rho^t(Z) = \sum_{X \in C} \rho^t(X, Z). \tag{15}$$

### 6.2.3 The Reputation Score

Once aggregated ratings for a particular agent are known, it is possible to calculate the reputation probability distribution for that agent. This also takes into account the base rate reputation score $a$ of all agents in the community. The reputation score is then expressed as:

$$\text{beta}(\rho^t(Z)) = \text{beta}(r + 2a, \ s + 2(1 - a)), \tag{16}$$

where

$$\rho^t(Z) = \left[ \begin{array}{c} r \\ s \end{array} \right].$$

However probability distributions, while informative, cannot be easily interpreted by users. A simpler point estimate of an agent's reputation is provided by $\mathrm{E}[\text{ beta}(\rho^t(Z)) ]$, the expected value of the distribution. This provides a score in the range $[0, 1]$, which can be scaled to any range (including, for example, '0% reliable to 100% reliable').

**Definition 5 (Reputation Score)** *Let $\rho^t(Z) = [r,s]'$ represent target $Z$'s aggregate ratings at time $t$. Then the function $R^t(Z)$ defined by:*

$$R^t(Z) = \mathrm{E}[\,\mathrm{beta}(\rho^t(Z))\,] = \frac{r + 2a}{r + s + 2} \qquad (17)$$

*is called $Z$'s reputation score at time $t$.*

The reputation score $R^t(Z)$ can be interpreted as a probability measure indicating how a particular agent is expected to behave in future transactions.

The base rate $a$ is particularly useful for determining the reputation score of agents for which the aggregated ratings have low confidence, e.g. because the agents have been idle for longer periods, or because they are new entrants to the community. It is interesting to note that in a community where the base rate $a$ is high, a single negative rating will influence the reputation score more than a single positive rating. Similarly, in a community where the base rate $a$ is low, a single positive rating will influence the reputation score more than a single negative rating. This nicely models the intuitive observation from everyday life where *"it takes many good experiences to balance out one bad experience"*.

### 6.3 Trust Reasoning

Subjective logic defines a number of operators [7, 22, 16], where some represent generalisations of binary logic and probability calculus operators, whereas others are unique to belief theory because they depend on belief ownership. Here we will only focus on the *discounting* and the *consensus* operators. The discounting operator can be used to derive trust from transitive paths, and the consensus operator can be used to derive trust from parallel paths. These operators are described below.

- **Discounting** [7] is used to compute transitive trust. Assume two agents $A$ and $B$ where $A$ has referral trust in $B$, denoted by $\omega_B^A = (b_B^A, d_B^A, u_B^A, a_B^A)$. In addition $B$ has functional trust in $C$, denoted by $\omega_C^B = (b_C^B, d_C^B, u_C^B, a_C^B)$. $A$'s indirect functional trust in $C$ can then be derived by discounting $B$'s trust in $C$ with $A$'s trust in $B$. The derived trust is denoted by $\omega_C^{A:B} = (b_C^{A:B}, d_C^{A:B}, u_C^{A:B}, a_C^{A:B})$. By using the symbol '$\otimes$' to designate this operator, we can write $\omega_C^{A:B} = \omega_B^A \otimes \omega_C^B$.

$$\begin{cases} b_C^{A:B} = b_B^A b_C^B \\[4pt] d_C^{A:B} = b_B^A d_C^B \\[4pt] u_C^{A:B} = d_B^A + u_B^A + b_B^A u_C^B \\[4pt] a_C^{A:B} = a_C^B \ . \end{cases} \qquad (18)$$

The effect of discounting in a transitive path is to increase uncertainty, i.e. to reduce the confidence in the expectation value.

- **Consensus** [7, 8, 9] is used to fuse two (possibly conflicting) beliefs into one. Let $\omega_C^A = (b_C^A, d_C^A, u_C^A, a_C^A)$ and $\omega_C^B = (b_C^B, d_C^B, u_C^B, a_C^B)$ be trust in $C$ from $A$ and $B$ respectively. The opinion $\omega_C^{A \diamond B} = (b_C^{A \diamond B}, d_C^{A \diamond B}, u_C^{A \diamond B}, a_C^{A \diamond B})$ is then called the consensus between $\omega_C^A$ and $\omega_C^B$, denoting the trust that an imaginary agent $[A, B]$ would have in $C$, as if that agent represented both $A$ and $B$. By using the symbol '$\oplus$' to designate this operator, we

can write $\omega_C^{A \diamond B} = \omega_C^A \oplus \omega_C^B$.

Case I: $u_C^A + u_C^B - u_C^A u_C^B \neq 0$

$$\begin{cases} b_C^{A \diamond B} = \dfrac{b_C^A u_C^B + b_C^B u_C^A}{u_C^A + u_C^B - u_C^A u_C^B} \\[10pt] d_C^{A \diamond B} = \dfrac{d_C^A u_C^B + d_C^B u_C^A}{u_C^A + u_C^B - u_C^A u_C^B} \\[10pt] u_C^{A \diamond B} = \dfrac{u_C^A u_C^B}{u_C^A + u_C^B - u_C^A u_C^B} \\[10pt] a_C^{A \diamond B} = a_C^A \end{cases}$$

Case II: $u_C^A + u_C^B - u_C^A u_C^B = 0$

$$\begin{cases} b_C^{A \diamond B} = (\gamma^{A/B}\, b_C^A + b_C^B)/(\gamma^{A/B} + 1) \\[6pt] d_C^{A \diamond B} = (\gamma^{A/B}\, d_C^A + d_C^B)/(\gamma^{A/B} + 1) \\[6pt] u_C^{A \diamond B} = 0 \\[6pt] a_C^{A \diamond B} = a_C \ . \end{cases}$$

where the relative weight $\gamma^{A/B} = \lim(u_C^B/u_C^A)$

The effect of the consensus operator is to reduce uncertainty, i.e. to increase the confidence in the expectation value. In case the subjective opinions are probability values ($u = 0$), Case II produces the weighted average of probabilities.

The discounting and consensus operators will be used for the purpose of deriving trust measures in the example below.

## 7  Example Derivation of Trust Measures

Transitive trust graphs can be stored and represented in a computer system in the form of a list of directed trust arcs with additional attributes.

This numerical example is based the trust graph of Fig.3. Table 1 specifies trust measures expressed as opinions. The DSTC Subjective Logic API[2] was used to compute the derived trust values.

Table 1: Direct trust measures of Fig.3

| Arc | Variant | Measure | Time |
|-----|---------|---------|------|
| $[A, B]$ | $r$ | (0.9, 0.0, 0.1, 0.5) | $\tau_1$ |
| $[A, D]$ | $r$ | (0.9, 0.0, 0.1, 0.5) | $\tau_1$ |
| $[B, C]$ | $r$ | (0.9, 0.0, 0.1, 0.5) | $\tau_1$ |
| $[C, E]$ | $f$ | (0.9, 0.0, 0.1, 0.5) | $\tau_1$ |
| $[D, C]$ | $r$ | (0.3, 0.0, 0.7, 0.5) | $\tau_1$ |
| $[A, B]'$ | $r$ | (0.0, 0.9, 0.1, 0.5) | $\tau_2$ |

A parser based on the algorithms of Fig.5 and Fig.7 can go through the arcs of Table 1 to construct the trust

---

[2]Available at `http://security.dstc.com/spectrum/`

network of Fig.3, and the corresponding canonical expression of Eq.(4). By applying the discounting and consensus operators to the expression of Eq.(4), a derived indirect trust measure can be computed.

- Case a:

First assume that $A$ derives her trust in $E$ at time $\tau_1$, in which case the first entry for $[A, B]$ is used. The expression for the derived trust measure and the numerical result is given below.

$$\begin{aligned} \omega_E^A &= ((\omega_B^A \otimes \omega_C^B) \oplus (\omega_D^A \otimes \omega_C^D)) \otimes \omega_E^C \\ &= (0.74, \ 0.00, \ 0.26, \ 0.50) \end{aligned} \quad (19)$$

The derived trust measure can be translated into a beta PDF according to Eq.(9) and visualised as a density function as illustrated by Fig.11
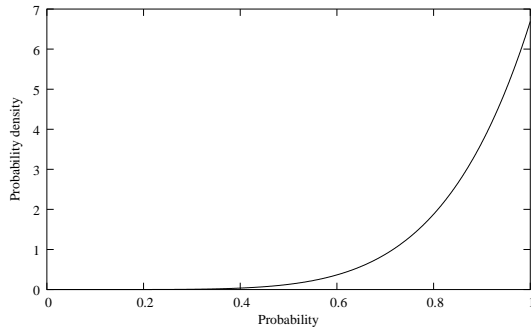


Figure 11: $\omega_E^A \equiv \text{beta}(6.7, \ 1.0)$

- Case b:

Let us now assume that based on new experience at time $\tau_2$, $A$'s trust in $B$ suddenly is reduced to that of the last entry for $[A, B]$ in Table 1. As a result of this, $A$ needs to update her derived trust in $E$ and computes:

$$\begin{aligned} \omega_E'^A &= ((\omega_B'^A \otimes \omega_C^B) \oplus (\omega_D^A \otimes \omega_C^D)) \otimes \omega_E^C \\ &= (0.287, \ 0.000, \ 0.713, \ 0.500) \end{aligned} \quad (20)$$

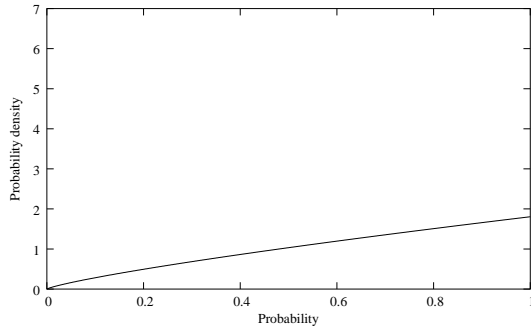Fig.12 below visualises the derived trust measure.



Figure 12: $\omega_E'^A \equiv \text{beta}(1.8, \ 1.0)$

It can be seen that the trust illustrated in Fig.11 is relatively strong but that the trust in Fig.12 approaches the uniform distribution of Fig.9 and therefore is very uncertain. The interpretation of this is that the distrust introduced in the arc $[A, B]$ in case (b) has rendered the path

($[A, B] : [B, C] : [C, E]$) useless. In other words, when $A$ distrusts $B$, then whatever $B$ recommends is completely discounted by $A$. It is as if $B$ had not recommended anything at all. As a result $A$'s derived trust in $E$ must be based on the path ($[A, D] : [D, C] : [C, E]$) which was already weak from the start.

## 8 Discussion and Conclusion

We have presented a notation for expressing trust networks, and a method for trust network analysis based on graph simplification and trust derivation with subjective logic. This approach is called Trust Network Analysis with Subjective Logic (TNA-SL).

Our approach is different from trust network analysis based on normalisation, as e.g. in PageRank and Eigen-Trust. The main advantage of normalisation is that large highly connected random graphs can be analysed while still taking all arcs into account. The main disadvantages of normalisation is that it is difficult to express negative trust, and that it makes trust measures relative, which prevents them from being interpreted in any absolute sense like e.g. statistical reliability. Neither PageRank nor EigenTrust can handle negative trust.

Trust network simplification with TNA-SL produces networks expressed as directed series-parallel graphs. A trust arc has the three basic attributes of source, target and scope, where the trust scope can take either the functional or the referral variant. This makes it possible to express and analyse fine-grained semantics of trust. Additionally, we have incorporated the attributes of measure and time into the model in order to make it suitable for deriving indirect trust measures through computational methods.

One advantage of TNA-SL is that negative trust can be explicitly expressed and propagated. In order for distrust to be propagated in a transitive fashion, all intermediate referral arcs must express positive trust, with only the last functional arc expressing negative trust. Another advantage is that trust measures in our model are equivalent to beta PDFs, so that trust measures can be directly interpreted in statistical terms, e.g. as measures of reliability. This also makes it possible to consistently derive trust measures from statistical data. Our model is for example directly compatible with Bayesian reputation systems [12, 25], so that reputation scores can be directly imported as trust measures. This rich way of expressing trust separates between the nominal trust value (positive/negative) and the confidence level (high/low), and also carries information about the baseline trust in the community.

The main disadvantage of TNA-SL is that a complex and cyclic network must be simplified before it can be analysed, which can lead to loss of information. While the simplification of large highly connected networks could be slow, heuristic techniques can significantly reduce the computational effort. This is done by ignoring paths for which the confidence level drops below a certain threshold, and by including the paths with the strongest confidence level first when constructing a simplified network. This also leads to minimal loss of information.

The approach to analysing transitive trust networks described here provides a practical method for expressing and deriving trust between peers/entities within a community or network. It can be used in a wide range of applications, such as monitoring the behaviour of peers and assisting decision making in P2P communities, providing a quantitative measure of quality of web services, assessing the reliability of peers in Internet communities, and evaluating the assurance of PKI certificates. Combined with subjective logic, TNA-SL allows trust measures to be efficiently analysed and computed, and ultimately interpreted by humans and software agents.

## References

[1] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Conference on Security and Privacy*, Oakland, CA, 1996.

[2] B. Christianson and W. S. Harbison. Why Isn't Trust Transitive? In *Proceedings of the Security Protocols International Workshop*. University of Cambridge, 1996.

[3] M.H. DeGroot and M.J. Schervish. *Probability and Statistics (3rd Edition)*. Addison-Wesley, 2001.

[4] C. Ellison et al. *RFC 2693 - SPKI Certification Theory*. IETF, September 1999. url: http://www.ietf.org/rfc/rfc2693.txt.

[5] P. Flocchini and F.L. Luccio. Routing in Series Parallel Networks. *Theory of Computing Systems*, 36(2):137–157, 2003.

[6] T. Grandison and M. Sloman. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3, 2000.

[7] A. Jøsang. A Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, June 2001.

[8] A. Jøsang. The Consensus Operator for Combining Beliefs. *Artificial Intelligence Journal*, 142(1–2):157–170, October 2002.

[9] A. Jøsang, M. Daniel, and P. Vannoorenberghe. Strategies for Combining Conflicting Dogmatic Beliefs. In Xuezhi Wang, editor, *Proceedings of the 6th International Conference on Information Fusion*, 2003.

[10] A. Jøsang, E. Gray, and M. Kinateder. Simplification and Analysis of Transitive Trust Networks (to appear). *Web Intelligence and Agent Systems*, 00(00):00–00, 2005.

[11] A. Jøsang, S. Hird, and E. Faccer. Simulating the Effect of Reputation Systems on e-Markets. In P. Nixon and S. Terzis, editors, *Proceedings of the First International Conference on Trust Management (iTrust)*, Crete, May 2003.

[12] A. Jøsang and R. Ismail. The Beta Reputation System. In *Proceedings of the 15th Bled Electronic Commerce Conference*, Bled, Slovenia, June 2002.

[13] A. Jøsang, R. Ismail, and C. Boyd. A Survey of Trust and Reputation Systems for Online Service Provision (to appear). *Decision Support Systems*, 00(00):00–00, 2006.

[14] A. Jøsang and S. Pope. Normalising the Consensus Operator for Belief Fusion. In *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence*, Sydney 2005.

[15] A. Jøsang and S. Pope. Semantic Constraints for Trust Tansitivity. In S. Hartmann and M. Stumptner, editors, *Proceedings of the Asia-Pacific Conference of Conceptual Modelling (APCCM) (Volume 43 of Conferences in Research and Practice in Information Technology)*, Newcastle, Australia, February 2005.

[16] Audun Jøsang, Simon Pope, and Milan Daniel. Conditional deduction under uncertainty. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, 2005.

[17] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, May 2003.

[18] Liberty-Alliance. *Liberty ID-FF Architecture Overview*. Version: 1.2-errata-v1.0. http://www.projectliberty.org/specs/liberty-idff-arch-overview-v1.2.pdf, 2003.

[19] Liberty-Alliance. *Liberty Trust Models Guidelines*. http://www.projectliberty.org/specs/liberty-trust-models-guidelines-v1.0.pdf, Draft Version 1.0-15 edition, 2003.

[20] G. Mahoney, W. Myrvold, and G.C. Shoja. Generic Reliability Trust Model. In A. Ghorbani and S. Marsh, editors, *Proceedings of the 3rd Annual Conference on Privacy, Security and Trust*, St.Andrews, New Brunswick, Canada, October 2005.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[22] Simon Pope and Audun Jøsang. Analsysis of competing hypotheses using subjective logic. In *Proceedings of the 10th International Command and Control Research and Technology Symposium (ICCRTS)*. United States Department of Defense Command and Control Research Program (DoDCCRP), 2005.

[23] Ronald L. Rivest and Butler Lampson. SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession, 1996.

[24] R. Wishart, R. Robinson, J. Indulska, and A. Jøsang. SuperstringRep: Reputation-enhanced Service Discovery. In *Proceedings of the* 28th *Australasian Computer Science Conference (ACSC2005)*, 2005.

[25] A. Withby, A. Jøsang, and J. Indulska. Filtering Out Unfair Ratings in Bayesian Reputation Systems. *The Icfain Journal of Management Research*, 4(2):48–64, 2005.

[26] WS-Trust. *Web Services Trust Language (WS-Trust)*. ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf, February 2005.