Trust management formal techniques and systems[*]

Guillaume Piolle[†]

december 2010

Abstract

The aim of this study is to provide an overview of the different ways to introduce trust management mechanisms in logic-based computing systems (especially multi-agent systems), in order to deal with security problems. This study includes a survey of recent publications in this research area, as well as a critical comparison between them. We try here to put a stress on their common aspects, their differences, the kind of logic paradigm they are based on, and the kind of issues they are likely to address well.

1 Introduction

In human societies, trust is a key notion in all kinds of relationships. It allows us to interact with other people in the safest way, even when we don't have all the necessary information about the subject of the communication, or about the communicants. In many cases, trust is a natural security layer in human organisations. For example, individuals who repeatedly behave in an anti-social or non efficient way are first distrusted only by their direct interlocutor, and then this "distrust" propagates itself and soon the whole community will be reticent to rely on such inefficient individuals, thus increasing the global quality of the collective work.

For several years now, efforts have been made in order to define this notion of trust in a formal way, to analyse it, to build logical models simulating trust in the most accurate manner. Such frameworks are being used in communication infrastructures; often including only a few aspects of what we commonly name "trust".

Trust-based systems and frameworks are often used to add a new security layer in collaboration issues such as knowledge sharing and management, resource sharing, access control, task delegating.

In this study we want to explore the different ways in which logicians, computer scientists and engineers deal with artificial trust, which notions are formally described, which ones are not, where are the strengths and potential weaknesses in all these works. From this overview of the state-of-the-art informal trust management, we want to identify the key notions that should be included in an efficient logical framework, and the best way to describe and use them in a formal way.

In part 2, we will introduce the various notions that appear to be useful when dealing with trust in a formal way, and discuss the different points of view that can be considered about trust and its lifecycle. In part 3, we will briefly present the different projects and theoretical works that we will study. In part 4, we compare more finely how each one of them deals with the different steps in the trust lifecycle (and how the lifecycles themselves can differ). In part 5, we will specifically focus on Sadri and Toni's work [11], trying to point out the possible evolutions of this trust management layer.

[*] This document is an author version of an Individual Study Option report, submitted in partial fulfilment of the requirements for the MSc Degree in Advanced Computing (research pathway) of the University of London and for the Diploma of Imperial College of Science, Technology and Medicine. This study was supervised by Dr Fariba Sadri (fs@doc.ic.ac.uk).

[†] Current contact address: guillaume.piolle@supelec.fr

2 Background

2.1 Definition of trust

Many definitions can be found for the notions of trust, including more or less subnotions. In [8], Grandison and Sloman give us a quite complete one, which should be quite easy to accept. They define trust as "a quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context". In parallel, distrust would be a "quantified belief by a trustor that a trustee is incompetent, dishonest, not secure or not dependable within a specified context". However, some people think that distrust should not be considered as a separate notion, and that one should only work with "trust" and "absence of trust" [3]. We will discuss this issue later.

In this study, we will mainly deal with trust in the context of multi-agent systems. Some of the works we are about to see only deal with access control. These systems usually are non-logic-based, and only include few of the sub-notions of trust.

In [4], Demolombe adds more sub-notions to this list of properties ("competence, honesty, security and dependability"). For him, trust is a belief concerning sincerity, cooperativity, credibility, vigilance, validity, completeness... He has a very complex model of trust, including these numerous sub-beliefs.

The notion of context is quite important: A cannot trust B in an absolute way. That would lead A to rely on B in some tasks for which B is not competent, motivated, or generally reliable. In fact A trusts B for a specific action or class of actions, or for a specific domain, either because B's actions have demonstrated that it is competent, honest... in that domain, or because B is "approved" for this action, having a certificate proving a specialisation, or the recognition of its authority.

Everybody seems to agree on the fact that trust is not symmetric in the general case: A can trust B for a given action, even if B doesn't trust A for the same action. Nevertheless, beyond this quite specific trust, one could imagine a wider notion, that could correspond to a symmetric trust: we could consider that A and B trust each other, without context. Here trust would be only a vague and general disposition of a community. It is quite uncomfortable to work with this notion, and in general, the people working in trust management consider only this commonly accepted asymmetry of trust.

According to Castelfranchi in [3], trust is divided in three aspects. These are interesting, socially speaking, even though they have the strong flavour of the "high care organisations" Castelfranchi works with. Trust would thus be:

- A mental attitude, a disposition: one can be "trusting" or "non-trusting". This notion allows us to define a symmetric binary relationship, and thus equivalence sets of trusting or non-trusting agents;
- A decision, an intention: the trust of A in B can be considered as a will to share knowledge with it, or to delegate a task to it;
- A behaviour, an act: this is how trust is built, and evolves. The trust of A in B influences its behaviour towards it, and the way it respond to its requests, but it also evolves accordingly to these actions.

An agent can also trust something that is not a cognitive agent, i.e. a simple resource on a network for instance. In this case, we would use a simplified model of trust, since the object of our trust may have no belief, no desire, but only more or less measurable and objective properties such as efficiency, performance, bandwidth, availability...

Sometimes, logicians and computer scientists working in the field of trust management seem to "aggregate" a wide set of vaguely related beliefs to this notion of trust, and end up by mixing the notions of believing and trusting. In some papers, like [3] , trust is not anymore this composite belief, but a multitude of little beliefs in various properties. This variety of definitions, not always very clear, can be quite confusing, nevertheless the definition given by Grandison and Sloman seems fairly able to describe the notions used in the other works studied here.

2.2 Definition of trust management

In [8], Grandison and Sloman define trust management as “the activity of collecting, codifying, analysing and evaluating evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships [...]”.

In fact, trust management seems to be commonly considered as the ways of including the previously defined notion of trust in a decision process. And this involves all the specific actions listed by Grandison and Sloman.

2.3 Initialisation and evolution of trust

In the field of multi-agent systems, trust is a belief, and often a composite belief, in the knowledge base of the agent. The agent often has various trust values, for each interlocutor in its environment.

At the beginning of the interactions between the agent and the environment, trust has to be initialised, or given a default value for each possible case. This initialisation can be totally static, for instance imposed by an administrator in a customisable part of the framework. In this case, it often concerns well-known third parties, more than general cases. After this possible static initialisation, the system always has a default behaviour for unknown agents or resources: they can be trusted by default, distrusted, or a neutral level can be defined.

After that initialisation, trust evolves, along with the interactions between the agent and its environment. It is influenced by how the transactions (knowledge sharing, task delegation...) go between agents, and by the recommendations the agent receives about other agents or resources. Basically, the success or failure of the interactions makes agent’s trust in its interlocutor increase or decrease, respectively. The recommendations, given by other agents or by central authorities, can influence the different trust values of the agent, depending of the level of trust the agent has in the source of the recommendation, and possibly other factors.

The rules that make trust evolve can often be configured (at least partially) by an administrator. It gives the possibility to define groups, roles and so on. Such role-based policies can also be part of the original framework, even sometimes with the notion of “authorised agents” [11], specialised in such or such task, that are mentioned in special policies in the agents’ trust engines.

According to all the papers considered in that study, trust absolutely has to be context-dependent. An agent trusts another agent only for a specific kind of task or goal. “Absolute trust” is very rare, even though some systems allow it. Sometimes [11] trust is also defined with a time reference, that allows the policy rules to deal with the history of events and interactions.

2.4 Composition of trust

According to Cristiano Castelfranchi [3], trust should be composed of several subbeliefs:

- **Competence belief:** agent A believes that agent B is able to do efficiently what A wants from it;
- **Willingness belief:** agent A believes that agent B wants (or needs) to do what A expect from it, because it helps it reaching one of its goals. This belief can be decomposed in several beliefs about the reasons of the other agent;
- **Dependence belief:** agent A believes that it has to rely on another agent because A cannot do it by itself (strong dependence) or that it is better for A that it relies on another agent (weak dependence).

Castelfranchi also differentiates different kinds of trust:

- **Blind trust**, which would be a “default” trust before any event on the system, and which would induce agents to initiate relationships with unknown entities;

- **Conditional trust**, which is the "classical" state of trust during the life of the agent. This conditional trust is likely to evolve, and can be subject to some sets of constraints or conditions (indicator thresholds, temporal conditions, internal beliefs. . .);
- **Unconditional trust** in a specific entity. Such a trust would probably be configured directly by an administrator, and would not be sensitive to successful/unsuccessful interactions, extern recommendation or any other source of evolution of the conditional trust. The interest of such a trust is that it is less expensive in terms of system resources: it needs less computation than a conditional evolution calculus.

In the same paper, Castelfranchi introduces, in addition to trust in someone (agent, resource), a trust in the process itself, or in the environment. For instance, let's say agent B has a very good trust value for the agent B, in the context of a task T. A and B are on a communication network, and the "environment trust structure" of agent A allows it to have different trust values for all the paths on the network. If the trust of the agent A in the path AB is not sufficient, A will not rely on B for the task T, in spite of its confidence in the capabilities of the agent B.

2.5 Using trust

When an agent has to rely on another entity in the environment (by delegating a task for instance), or more generally to take a decision involving another entity (like providing an access token), the decision-making process uses the trust values in its knowledge base.

Generally, the decision-making process is led by policy rules, that impose some values to some specific variables of the trust data structure.

The policy rules can often be configured by an administrator. Sometimes, they are subject to specific constraints [11], allowing more precision and customisation in the process.

The general principle of decision-making is quite similar in all works. Nevertheless, this is a part of the trust lifecycle that differs quite much, in its details, from one framework or system to another. For instance some systems include a risk or cost evaluation engine in the process.

3 Some trust management frameworks

We will see here a quick overview of all the systems and frameworks considered in this study, with for each one, the kind of formalism used, the general principles, and if possible the kind of applications it is designed for.

3.1 Sloman's SULTAN

Corresponding papers: [8, 9].

SULTAN is a trust management framework intended to be "a component within a set of on-line management tools for distributed systems" [8]. It is composed of a "specification editor" that allows the administrator to define the policy rules for trust specification, a trust analysis tool based on Sicstus Prolog, and a "risk service". The system is designed to support queries from authorised users, concerning trust values in a given context. In fact it is a centralised trust engine, adapted to an Internet use. The original objective was to design something better than the traditional PKIs, something which would allow "the analysis of trust relationships, the use of auxiliary factors, such as risk and experience, in trust decision making, and the fact that trust is a dynamic concept that evolves with time" [9].

In this system, the trust analysis engine works together with the risk service. Here risk is viewed as a "probability of a failure with respect to the context of the interaction". Thus risky actions would necessitate a superior level of trust. The trust engine works with a quantified but unique notion of trust, context-dependent, dependent on constraints, and supporting a recommendation system.

The system also includes the notion of (quantified) distrust.

One of the objectives of this system was to do better than already existing frameworks (cited in [9]: Jøsang's subjective logic, Jones and Firozabadi's model, Rangan's model), which had, according to Grandison and Sloman, the following problems (quoted from [9]):

- Their underlying assumptions make them non applicable to the distributed framework;
- There is no associated tool support;
- They address a subsection of the trust management problem.

3.2 Sadri and Toni's work

Corresponding papers: [10, 11].

This work is a trust management layer designed to improve the planning process of an agent, developed in parallel with the Knowledge-Goal-Plan agency model [10]. It is a quite simple system, based on the abductive logic (with integrity constraints), which is the core of the agent planning engine. In the sixth part of this study we will try to identify possible improvements to this trust management system.

In the agent planning process, from an "abductive logic" point of view, the observation of the abductive program is the goal plus the observations of the agents, the explanation computed gives the actions to be undertaken by the agent in order to reach the goal, provided the observations.

This process is included in an overall observe-reason-act cycle defined in [10] for the KGP model of agents.

Currently this work includes conditional trust, the notion of time (and the precedence of actions necessary to the planning), a recommendation system, a recognition of the specific authority of certain entities in certain domains.

3.3 Castelfranchi's framework

Cristiano Castelfranchi's trust management framework [3] is meant to be used in a "high care" agent organisation, a multi-agent system based on the BDI model where members form a "supporting social network", trying to improve trusting relationships. The agents try to help the other ones being comfortable with asking help: they encourage low-trust communications, they're indulgent towards errors, and they all care about a common goal.

Here trust is mainly used to regulate knowledge management and sharing.

Castelfranchi proposes in his paper a way to deal with trust, not linked with any kind of logic, but that could be "translated" in several logical paradigms.

Here, "reciprocity, reputation and altruism" are the concepts over which is based the "knowledge market" of the system. Frequent communications are encouraged, in order to improve trust between agents.

In this kind of network, many beliefs are shared by part of, or the whole agent community, in particular belief about the value of knowledge sharing, or the goal of being well-evaluated and trusted.

3.4 The SECURE project

Corresponding papers: [2, 5, 6, 12].

SECURE is a European research project. It is a trust and logic-based Global Computing framework, including a trust-based access control framework, OASIS [5], which can be considered as a centralized trust management framework: a central access control manager uses a "trust calculator" and a "cost analyser" to provide (or not) access tokens to agents (here called "principals") asking for access or action rights corresponding to their roles.

This framework is partly based on the theory of domains and fixpoints, which are used to make trust evolve dynamically along communication between agents. We will not go further into the specific use of domains and fixpoints in this paper.

In the SECURE project much work has been done to build a “trust structure” able to integrate all the observations and communications of the agent, and to compute trust-related values at any required level.

3.5 Demolombe’s work

Corresponding paper: [4].

In his work, Demolombe does not present an actual trust management framework, but a way to define many components of the trust of an agent. He seems to work with modal and deontic logic.

Demolombe defines a number of “epistemic properties”, which basically define the qualities of an agent or entity, such as “sincerity” or “credibility”, and then he gives a formal definition of trust, in relation with all these properties. He also uses logic to reason about the existing links between all the components of the trust belief he has defined.

Demolombe also present some very interesting (and theoretic) considerations on the gradation of trust and beliefs, and on conditional trust.

3.6 Other logic-based papers

Corresponding papers: [1, 13].

These two papers are position papers, they suggest plan of research, more than actual improvement in the domain. They don’t bring anything really new or original compared to the other ones.

3.7 Non logic-based frameworks

Corresponding paper: [7].

The survey made by Grandison and Sloman in this paper mainly refers to non logic-based trust management systems. What we see at first sight is that each one of them only addresses one or two of the aspects of trust management.

The first family of these systems are the public key infrastructures. X509 (strictly hierarchical) and PGP (distributed and non hierarchical) are quoted in [7]. In fact this kind of infrastructures only gives us a certificate of identity for an entity, and no information about its reliability. From this point of view, PKIs could hardly be considered as trust management tools. In fact they would be integrated with much benefit in a larger framework, dealing with all the aspects of trust and relying on the PKI for the authentication and identification.

Platform for Content selection (PICS), developed by the World Wide Web Consortium, is another of these systems. In fact it is only the specification of a meta-document system that allows content providers to append a tag to their documents, and client software developers to deal with the information in these tags. REFEREE (Rule-controlled Environment for Evaluation of Rules and Everything Else), another system mentioned in this paper, is an example of client application filtering web documents using the PICS labels.

The W3C PICS documents include rule syntax, but the implementation is the developer’s responsibility. The information (the labels) that may be contained in the tags are totally up to the developers: any customised information can be added, including numeric values. In fact the recommendation only describes the syntax. All the labels may be referenced in a “rating-system” that would be the PICS equivalent for a XML DTD.

Another interesting system referenced in this paper is the AT&T PolicyMaker and KeyNote (KeyNote is the successor of PolicyMaker). PolicyMaker is a trust management application that allows an administrator do define rights attached to a public key. It is built like a query server, managing local policies (rules defined by the administrator), received credentials (third-party certificates). In fact this system focuses on access control and service provision. The policy rules are of the following form:

```
policy
    ASSERTS public_key
    WHERE filter_returning_an_authorisation
```

Where `policy` is the source of the policy (which is actually "policy" on the local system), `public_key` is the target public key, and `filter_returning_an_authorisation` is a script (AWKWARD, Java or Safe-TCL) checking the properties of the public key and returning a specific right. A list of the main disadvantages of these frameworks has been established in [9] by Grandison and Sloman. They are criticised for the following:

- The responsibility of checking the conditions of the establishment of a trust relationships is entirely the application developer;s concern;
- The relationships are assumed to be monotonic;
- These solutions do not learn from the information available to them;
- The emphasis is on access control decisions rather than general analysis of trust.

In the rest of the ISO we will concentrate on the logic-based approaches, which appear to address more issues in the field of trust management.

4 More detailed description of the different approaches

In this section, we will recall the concepts introduced in the background section, and for each one we will detail the way the different trust frameworks deal with it.

4.1 Initialisation of trust

4.1.1 Sources of trust

In all these works, at the initialisation of the system the trust values can only come from these three possible sources:

- Customised trust establishment rule given by an administrator;
- Default trust level (different for trustful or distrustful agents, or in quantified or non quantified systems);
- Trust generated by the analysis of a prior recommendation.

Yet, some of the works do not consider all of them, for instance the purely theoretical papers like [3] and [4] rarely consider the opportunity for a human administrator to define or "force" a trust value, whereas all the "practical" works include this possibility.

In SULTAN, trust is initialised by logical "trust policies" defined by the administrator, of the following form:

```
PolicyName:  trust(Trustor, Trustee, ActionSet, Level) <- ConstraintSet;
```

We can see that trust is actually defined in a context ("ActionSet" parameter), quantified ("Level" parameter, that can be either a numerical value or a customised label), and conditioned by a set of constraints. In the same logical formalism, one can specify recommendation policies, in relation with an action set, some constraints, and with a given level.

In the work of Sadri and Toni [11], one could also find the notion of authorisation, like in the following example rules, quoted from [11]:

```
authorised(X, give_advice(msc_course), T) <- msc_director(X, T)
authorised(X, give_advice(hotel_booking(Hotel)), T) <- receptionist(X, Hotel, T)
```

One can remark the timestamp, interesting notion that could allow specifically limited authorisation, for instance. With the same idea, a whole role-based trust component can be defined for a model or a specific agent. With these authorisations, specialisations emerge from the population of the agents, and lead to a kind of unconditional trust from other agents, with respect to the corresponding task.

Nevertheless, the notion of authorisation (certification?) seems local to the agent: an agent has or doesn't have the authorised (...) policy in its trust engine. One could think that a notion of certification authority would be interesting here.

4.1.2 Quantified trust

All the works, excepted Sadri and Toni's system [11] have a notion of quantified trust, even though this notion may vary between the papers.

In [3], Castelfranchi has a very precise opinion of how to work with this quantification, which is "a function of the subjective certainty of the relevant beliefs". He assumes that "the quantitative dimensions of trust are based on the quantitative dimensions of its cognitive constituents". Here the degree of trust DoT_{XY_τ} of an agent X about an agent Y on a task τ is computed like this (quoted and adapted from [3]):

$$DoT_{XY_\tau} = DoC_X[Opp_Y(\tau)] \times DoC_X[Ability_Y(\tau)] \times DoC_X[WillDo_Y(\tau)]$$

Where:

- $DoC_X[Opp_Y(\tau)]$ is the degree of credibility of X's belief about Y's opportunity of performing τ;
- $DoC_X[Ability_Y(\tau)]$ is the degree of credibility of X's belief about Y's ability/competence to perform τ;
- $DoC_X[WillDo_Y(\tau)]$ is the degree of credibility of X's belief about Y's actual performance.

In this expression the "actual performance" for instance is a dimension defined precisely in the kind of multi-agent system Castelfranchi works with, but one could adapt this expression to one's own components of the trust belief, like belief in Y's competence, honesty, security and dependability, or whatever list of qualities.

When using this calculus principle, the degrees of credibility of the beliefs also have to be defined. One could use for instance the notion, given by Demolombe in [4], of "belief, strong belief and knowledge" that could introduce a three-value quantification for the source beliefs, and thus a 27-value quantification for the overall trust.

Demolombe, in [4], introduces another interesting way of quantifying trust (what he calls "graded trust"). For him, the components of trust are beliefs in logical implications. For instance the trust of the agent A in the sincerity of the agent B is the belief (or knowledge):

$$K_a(I_{b,a}p \rightarrow B_b p)$$

That means: A knows (believes) that if B informs A that p, then B believes that p. In a general way:

$$K_a(\Phi_b \rightarrow \Psi_b)$$

For this belief, the strict implication means that the sets of worlds where Φ_b is true is totally included in the set of worlds where Ψ_b is true. In order to deal with a level of inclusion of those two sets, Demolombe introduces a quantified entailment operator: $\Rightarrow_i$, where i is the level of inclusion of the two sets. Then he defines the level of trust by using the modified precedent belief:

$$Tprop^i_{a,b}p = K_a(\Phi_b \Rightarrow_i \Psi_b)$$

Where $Tprop^i_{a,b}p$ is the level of trust of the agent A in the agent B for the property $prop$ (sincerity for instance), concerning the fact p, and i is the level of inclusion of the two sets of worlds, or the level of entailment of the two properties.

4.1.3 Quality of information

In the SECURE project, and more precisely in OASIS [5], a quite original notion is introduced. Here trust information is rated not only for its pure trust value, but also for its information value. SECURE uses a very complex layer-based model for dealing with trust, and we will not detail it here, but the interesting idea is that the trust value has two orderings defined on it, trustworthiness (pure trust) and information (used to precise how much evidence was used by the trust engine to compute the trust level).

This idea is quite interesting, in particular in the case where agent are naturally trustful. Let's imagine an agent A, trustful (gullible?), and an agent B, unknown from A, who actually is an evil agent dedicating its existence to the failure of A's goals... Let's say A has a task to perform, and has to delegate it to another agent, but he has for no agent a better trust value than its default trust. So he is likely to contact B, just like any other agent, and the task may not be completed properly. But if the agent A has an information value associated with its trust value, he will be able to know that another agent, C, if having the same trust value than B, has a better information value, because A has already worked with it in the past. So A will prefer rely on C rather than on B, and A's task is saved by the information value!

4.1.4 Defining distrust?

The large majority of the quoted works define the notion of distrust (which may be quantified, as well as trust). This seems quite natural to give agents the ability to distrust an inefficient or corrupted entity. This notion has a strong socio-cognitive meaning.

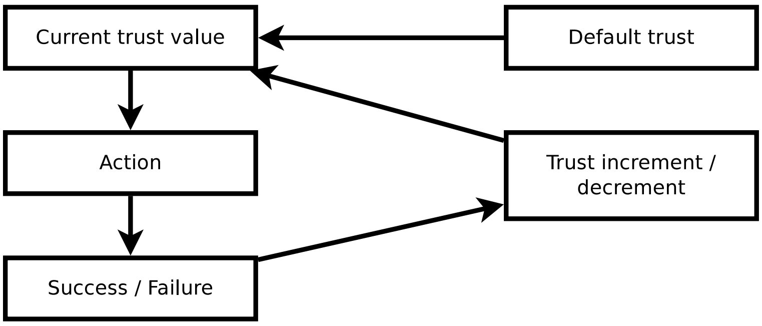
Nevertheless, some people like Cristiano Castelfranchi [3] think that trust should only be a positive belief, since the introduction of a negative trust (distrust) would actually induce the most dishonest agents to get round the system. In effect, imagine a trust management system where you have defined distrust, and where you have a default trust level. Imagine an agent A, characterised by its inefficiency or more generally unreliability, but whose goal is to be trusted by other agents (or at least he has some interest in that). Such an agent, having very bad trust values (high distrust) with all the other agents of the network, might be interested in creating a new identity (whatever meaning it has on the system) and to appear as a new agent, benefiting from a brand new default trust value. In that case the trust value of the entity controlling the agent would have unduly increased, and it would have a better trust value that other agent, maybe not very reliable but more than the former agent A.

On the other hand if your system has the same neutral default trust value, but no distrust defined, then the default trust value is the worst trust value an agent could ever had, so there is no inducement in hiding or changing one's identity, and this possibility of getting round the system disappear. In such a system (typically the high care networks Castelfranchi works with), agents are invited to initiate low-trust transaction with new agents, and all is made to make the agents "confidents" towards the society (i.e. given the parameters of the trust-based communication, communicating, relying on other agents is not against one's goal of being trusted by the network).

4.2 Evolution of trust

As we have seen sooner, trust values basically evolves along the succession of transactions between the agent and other entities, and with received recommendations. This is just the general principle, commonly shared by all the authors: when an agent A performs a transaction involving another agent B, if the behaviour of the agent B tends to help the transaction succeed, trust of agent A in agent B increases after the transaction. If the behaviour of the agent B tends to lead to the failure of the transaction, trust of agent A in agent B decreases. Concerning the recommendations, depending on the formalism they can be only positive, or either positive or negative. The way they influence trust is generally conditioned by the local policy of the agent.

The notion of "trust lifecycle" is often read in all those papers. In effect, trust is updated through a cycle:



This one is the most simple trust cycle possible: at the beginning trust is initialised with default trust. When an action is to be made, the current trust values are used in order to decide whether to rely on another agent or not, and on which agent. From the success or failure of the action, the system computes a modification factor for the concerned trust values, and applies it, thus modifying the current trust values.

More complex cycles can also involve risk management (like in Sloman's SULTAN, [8]), a computation of the value and/or cost of the action (Castelfranchi [3]), or application-specific information.

For Toni and Sadri [11], this notion of trust evolution has a quite different meaning, for trust is not quantified in their system. So when the agent has to perform an action, the trust it has in another agent is a direct function of the past actions: for instance he will consider it trusts the other agent only if it already relied on it in the past, with a successful conclusion, a minimum number of times. This is an example of how the local policies of the agents allow defining that (quoted and adapted from [11]):

```
trust(maria, X, T, "gift") ←
    tell(X, maria, "ok, I will give you G1 at time T1", T’),
    tell(X, maria, "ok, I will give you G2 at time T2", T’’),
    do(X, "deliver G1 to maria", T1),
    do(X, "deliver G2 to maria", T2),
    G1 ≠ G2.
```

In fact here, the agent Maria trusts another agent X if the sufficient condition of the agent X having fulfilled two contracts in the past is true. The "gift" argument is here to precise for what kind of task the agent X is trusted. In a general way, one could customise such rules to precise one's own demands: how many times the agent has been successful in the past? How many times did he fail? How long ago? Such a system also gives the basis for a notion of prescription or such complex behaviours. The management of recommendations is specified in the same kind of way, adding other sufficient conditions to trust:

```
trust(maria, X, T, Task) ←
    tell(Y, maria, recommended(X, Task), T1),T1 ≤ T,
    trust(maria, Y, T)
```

In this particular policy, we see that the only condition to trust another agent on the basis of a recommendation, is that we trust the source of the recommendation (at the present time though, not at the time of reception of the recommendation).

Some integrity constraints (and extension to the abductive logic, used in [11]) can also influence in some way that establishment of trust. For instance the following example tells that the agent maria trusts another agents if it is honest, and that honesty is not compatible with having been in prison before:

```
trust(maria, X, T, Task) ← honest(X, T).
honest(X, T) and in_prison(X, T’) and T’ ≤ T → false.
```

In the OASIS part of the SECURE project [5], the pure recommendation system is replaced by a credentials system. For memory, OASIS is a centralised trust-based access control manager.

The agent, here name principal, sends its request to the access control manager, along with "a list of credentials, which may include signed trust-assertions (recommendations) from other principals, and/or a list of referees whom the trust calculator may wish to contact for recommendations". This is a richer system, much inspired from the human administrations for instance, and which appear very natural and intuitive. This system also supports the notion of roles, based on those credentials. In OASIS credentials (individual or role-based) can be short-lived (delivered with a TTL) or persistent. OASIS also implements the notion of privilege delegation, which would be the superior level for the recommendation principle.

In the SULTAN framework, a monitoring engine is dedicated to observation of transactions, and modifications in the trust data structures [9]. It retrieves, computes and uses information about "up-to-date risk, experience and system state information". This monitoring system is the part of the framework that allows trust evolution.

In the SULTAN framework, the recommendation emission is subject to policy rules similar to the classical trust policies:

```
PolicyName:  recommend(Recommendation, RecommendedAgent, ActionSet, Level)
    → ConstraintSet;
```

4.3 Composition of trust

According to Grandison and Sloman's definition of trust [8], the trust belief should be composed of four sub-beliefs, which would be:

- Belief in the competence of the trustee;
- Belief in the honesty of the trustee;
- Belief in the security of the operation with the trustee;
- Belief of the trustor's dependability on the trustee.

Castelfranchi [3], him, uses beliefs in Competence, Willingness, and Dependence, where willingness is a belief in the necessity (or advantage) for the trustee to do the expected action. The trustor believes in the trustee's willingness for an action a, iff it trusts that the action a helps the trustee to reach one of its goals.

In [4], Demolombe presents a very complete set of beliefs that are, according to him, actual components of the socio-cognitive notion of trust, translated in his logical formalism. One could say that there are too many of them, and that some of them are not linked with the notion of trust, and actually in Demolombe's paper the limit between trust and general belief is sometimes confusing. Anyway the way he presents those beliefs is very interesting, and one could pick up the beliefs one is interested in for one's own application.

First Demolombe defines the different qualities of the potential trustee, on which will be based the different sub-beliefs. Each quality is a logical implication. In the following, B is the potential trustee, A is the trustor, p is a fact or logical proposition. I denotes an information given, B a belief.

- **Sincerity.** Anything the trustee communicates is believed by the trustee: $I_{b,a}p \rightarrow B_b p$;
- **Cooperativity.** The trustee communicates any information that is in its belief base: $B_b p \rightarrow I_{b,a}p$;
- **Credibility.** The beliefs of the trustee are well-founded: $B_b p \rightarrow p$;
- **Vigilance.** Any fact or proposition that actually holds is believed by the trustee: $p \rightarrow B_b p$.

Two complementary qualities can be defined, **Completeness** and **Validity**. Completeness means that the trustee informs the trustor about any fact or logical proposition that actually holds. Validity means that all that is communicated by the trustee actually holds. We can see that Completeness is the conjunction of Vigilance and Cooperativity, and Validity is the conjunction of Sincerity and Credibility.

After that, Demolombe defines trust in each of those qualities with the following:

$$\begin{aligned}Tsinc_{a,b}(p) &= K_a(I_{b,a}p \rightarrow B_b p)\\ Tcoop_{a,b}(p) &= K_a(B_b p \rightarrow I_{b,a}p)\\ Tcred_{a,b}(p) &= K_a(B_b p \rightarrow p)\\ Tvigi_{a,b}(p) &= K_a(p \rightarrow B_b p)\end{aligned}$$

As before, one could also define trust in the complementary qualities that are Completeness and Validity.

4.4 Using trust

As we have said all along this study, trust values are used to decide whether or not to rely on another entity for a specific action, and on which entity to rely on. Trust tells us who to contact for information, to which agent to delegate an action.

One or the most basic possible ways is proposed by Sadri and Toni in [11]. The simplest manner to deal with trust is: I trust you; it is a sufficient reason for me to ask you something. Such a rule would look like that:

```
request(maria, X, Action, T) ← trust(maria, X, T, Action).
```

Of course in Sadri and Toni's system such rules can be made more complex and customised, and for each specific action one could add specific conditions to basic trust.

In the SULTAN framework [8], like in other trust frameworks used as decision support tools, administrators make queries to the Analysis tool in order to retrieve some trust related information. Special kinds of queries (managed by the "source analysis") can force the trust analysis engine to ignore the constraint sets specified by the policies. Then the analysis tool just answers with the corresponding trust value. The queries are of the following form:

```
Query(Variables, Conditions, ResultSet).
```

Such trust information can then be used to configure the security of various network infrastructures, or to decide actions, to allocate access rights...

Trust is also used to generate recommendations. Classically, in the trust engine of an agent (or of a centralised trust server), we have policy rules defining the sufficient and/or necessary conditions in order to answer to a recommendation query. If the trustor trusts enough the trustee, he may answer positively to a recommendation query from another agent. In some systems (where distrust is defined), recommendations can also be negative. For instance in [8] and [9], such rules look like that:

```
PolicyName:  recommend(Recommendor, Recommendee, ActionSet,
   Level) → ConstraintSet ;
```

Here are some examples given in [8], that illustrate quite clearly the general principle of conditional recommendation:

```
R1:  recommend(BestShopper, Sainsbury,
    buy_products(Sainsbury), HIGHREC) ←
    BasketCost(Sainsbury) < £40;
```

Bestshopper recommends Sainsbury to provide a `buy_products` service with high confidence if the cost of a standard basket of products at Sainsbury is less than £40.

```
R2:  recommend(John, Fred, installOS(staffPC), -50);
```

John does not recommend Fred at a medium level (if the level range is [-100; 100]) to perform `installOS` on `staffPC`.

```
R3:  recommend(Morris, ICStudents,
    design_software(ICStudents), HIGH) ← trust(Naranker,
    ICStudents, program(ICStudents), MEDIUM);
```

Morris highly recommends ICStudents to provide a `design_software` service if Naranker reasonably trusts them (at MEDIUM level) to program.

4.4.1 Trust and risk, cost, utility

Trust is quite often used in parallel with risk or cost. In [8], Grandison and Sloman define risk as a "probability of a failure with respect to the context of the interaction". In general systems that work with trust adopt one of these two solutions:

- Risk (or cost) is defined by local rules, written by an administrator in respect with application-dependent specific needs, in the customisable part of the agent;
- Risk is calculated by a dedicated layer in the architecture of the agent model. This risk engine provides values for each action, at the decision-making step.

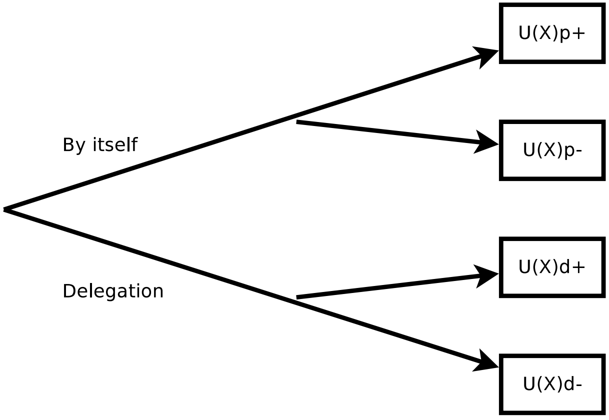
When working with risk, as in the SECURE project [5], the general principle is that the level of trust required for an action, depends on the risk calculated for this action. The higher the risk, the stronger the demanded trust.

In the SULTAN framework [9], risk is associated to the notions of "maximum allowed loss" and "acceptable risk" threshold, what allow very detailed queries and results. The risk calculation engine also includes a list of common risks, such as "receipt of malicious code", "refusal to produce goods", "service failure", "theft of information", "fraud"...

In [3], Castelfranchi works with utility functions, rather than risk values, and proposes a formal way to make the related decisions. In his papers he assumes that the agent X can calculate a utility function $U(X)_{sc}$ for each scenario sc. In the present case, he considers that the possible cases for sc are the following:

- $p+$: X does the action by itself and succeeds;
- $p-$: X does the action by itself and fails;
- $d+$: X delegates the action to another agent and it is a success;
- $d-$: X delegates the action to another agent and it is a failure.

And thus for each of these scenarios, he calculates a utility function:



If we consider that $DoT_{XY}\tau$ is the trust of the agent X in the agent Y properly performing the action τ, then the agent decides to delegate the action rather than doing it by itself if and only if:

$$\begin{aligned} &DoT_{XY}\tau \times U(X)_{d+} + (1 - DoT_{XY}\tau)U(X)_{d-} \\ &\quad > DoT_{XX}\tau \times U(X)_{p+} + (1 - DoT_{XX}\tau)U(X)_{p-} \end{aligned}$$

The same process can be used to choose between several possible agents.

4.4.2 Trust used in knowledge sharing

In [3], Castelfranchi deals with trust in the context of knowledge sharing. With this point of view, he defines a huge list of new beliefs that should be used, according to him, to define trust when working in knowledge sending and accepting.

The trustor should consider its beliefs in the trustee's understanding ability, appropriate use of the knowledge, attitude evolution toward the trustor, trust toward the trustor, needs, reputation, honesty, dependence, will to be well-evaluated and trusted by the trustor, its beliefs in the importance of the information, the necessity of the information with respect to a given goal, its trust toward the sources of the information, and its belief about the advantage of knowledge sharing. And in the case of accepting knowledge, the trustor should consider its beliefs about the source (is it well-informed, competent, trustworthy, not deceiving, not hostile. . .) and its beliefs about the information (is it useful?).

Castelfranchi mentions more than 20 sub-beliefs concerned by trust-based knowledge sharing! In that section once again the limit is not clear about simple beliefs, and trust-related beliefs, and one could say that some of these have nothing to do with trust.

4.4.3 Trust networks

Castelfranchi, still in [3], also introduces the notion of "trust networks". It is a way to represent the trust relationships in a multi-agent system. In fact he defines a graph, where the nodes are the agents, and the weights of the links are related to the trust values between the two agents. Due to the non-symmetric nature of trust, he defines unilateral, bi-directional and unbalanced arcs. He ends up quite naturally with giving more weight to the bi-directional ones.

He seems to use these trust networks to identify the "trust channels" in the networks, the relationships that are more likely to be used and enforced, should an inter-agent transaction be made. He introduces the "first and second bronze predictions" (quoted from [3]):

- The larger the channel, the larger the flux of knowledge and its rapidity;
- The more channels are bilateral and balanced, the more uniformly shared knowledge is.

It is not very clear in this paper what such trust networks are really useful for. In the case of Castelfranchi's "high care" systems, maybe the interesting point is that they give an idea of how well the information is spread on the network, or which agents are isolated from the other ones and from the information flux.

One could also suggest that those trust networks could be an interesting tools in making previsions in order to perform effective load balancing and network management, at an inferior abstraction level.

5 Possible evolutions to Sadri and Toni's framework

Sadri and Toni's work, presented in [11], has a few original features, like the introduction of the notion of time, which allows very powerful and complex policies, and the notion of authorised or expert agent that is only vaguely mentioned in the paper. After having had this overview of these different formal ways to deal with trust, maybe we could suggest a few interesting notions that could be added to this specific work. The purpose of this section is more providing general directions than actual improvements.

5.1 Friendship and unconditional trust

In the paper, we see the notion of "friends", but it is not very clear whether it is only an example predicate or a notion integrated in the agent model. It seems to be only a user-defined predicate, but if it were a built-in one, the interesting notion of "unconditional trust" could be based on it.

Nevertheless, the number of agents one is likely to trust unconditionally is quite little, and the designers of the systems should define whether unconditional trust would be related to a specific action or set of action or not. If not, the notion would lose much of its utility. Anyway the most significant advantage of unconditional trust is that it is cheaper in terms of system resources, but this advantage would be less evident in the case of abductive logic, it would depend of the implementation, and the selection rules of the abductive logic engine.

5.2 Decomposition of trust in several beliefs

In [11], trust is a unique predicate. An agent trust another agent or not, but in fact it is impossible to know why! It would be an interesting idea to introduce more trust predicates, for instance in relation with the different qualities proposed by Grandison and Sloman in [8]:

```
trust_competence(Trustor, Trustee, T, Action) ← ...
trust_honesty(Trustor, Trustee, T, Action) ← ...
trust_security(Trustor, Trustee, T, Action) ← ...
trust_dependability(Trustor, Trustee, T, Action) ← ...
```

With such precise trust-related predicates, an agent is able to know what qualities of the trustee it can rely on, since maybe for one kind of specific action, he will only need a subset of these notions. In particular, `trust_dependability` will tell it whether he needs or not to rely on another agent: maybe it will be more efficient and useful to do the task by itself. This should be investigated with respect to the cognitive model of the agent (for instance in Castelfranchi's high care networks, either `trust_dependability` would often hold, or the agents would rarely need it).

5.3 Quantified trust with information level

One could point out that this trust management layer is the only work not to deal with quantified trust. Here we have a "all or nothing" trust that looks a bit simplistic.

Since there is currently no notion of trust quantification, it may be a good idea to introduce by the same way a mean to quantify the information, like in OASIS. Because of the nature of the abductive logic program, maybe these two variables will not be bound at any time if they are numeric, but at least in the decision-making process one could obtain bounds for it. But it could also be more adapted to define labels for the quantification, for trust as well as for information. The following examples are only here to propose a way of dealing with such values, without making any suppositions on the actual rules.

```
trust(trustor, trustee, Time, Task, HIGH_TRUST, HIGH_INFO) ← ...
trust(trustor, trustee, Time, Task, DEFAULT_TRUST, NO_INFO)
    ← \+ any_transaction(trustor, trustee, T).
trust(trustor, trustee, Time, Task, MEDIUM_TRUST, InfoValue)
    ← \+ recommended(trustee, Task), InfoValue > MEDIUM_INFO.
```

5.4 Recommendation management

In this paper, we only see a few allusions to recommendations, in the form:

```
recommended(Trustee, Task).
```

It would be a richer notion if the source of the recommendation would be mentioned, with the date of the recommendation and maybe the duration or expiration date:

```
recommend(Source, Trustee, Task, Time, Expiration).
```

That way, the agent receiving the recommendation can deal with it differently depending on the date of reception, the trust it has in the source…

6 Conclusion

During this study we have seen that there are many different points of view on something as basic as the simple definition of trust. In the field of logic-based computing, everybody seems to agree that it is a belief, in most of the case a multiple belief in various qualities of the trustee. But everyone has his own set of qualities on which he bases his own definition of trust. For some people it is only a simple belief or predicate, for other ones it can includes dozens of sub-beliefs.

The different frameworks also have different levels of sophistication in the decisionmaking process. For a few of them, it is a centralised entity that decides who is trusted or not, and for what kind of activities. For the rest, every agent in the system has a personal "opinion" on every other one, and makes its own decisions by itself on the basis on the information it has.

Each of the works here has original notions, be it quantification of information, a way to deal with risk or value management, timestamps... and anyone wanting to build such a system should analyse those improvements, and decide whether or not they are suitable for his own application.

Some of the works [3, 4] provide more a theoretical study on trust than an actual working system, and maybe they are the most useful ones, for they study all implications of their model, and provide theoretical notions that could be (partially or as a whole) introduced in one's specific trust management system. This is the beginning of such a work that was intended to be done for [11] in the previous part of the study.

We can also see the advantage of formal techniques, and in particular the logic-based ones, which provide validation tools for the agent models [1], and that allow trust management model that are not application-dependent. The logic-based models studied here include the entire trust management layer, unlike the non-logic frameworks described in [7] for instance, that often rely on the client application developer, which seems quite dangerous when introducing a security layer.

We could also mention a problem evoked in [8], which is since all these trust models are different and incompatible, it is currently difficult to build a framework able to accept different kinds of agents. This problem is not specific to trust management of course! Such a framework, in order to allow the agents to communicate, would probably decrease the level of security by ignoring some concepts. But anyway, since these specific concepts are often field or application-dependent, maybe communication between agents coming from different systems is not a very relevant issue.

References

[1] M. Butler, M. Leuschel, S. Lo Presti, and P. Turner. The use of formal methods in the analysis of trust. In *Second International Conference on Trust Management (iTrust 2004)*, pages 333–339, 2004.

[2] M. Carbone, M. Nielson, and V. Sassone. A formal model for trust in dynamic networks. Technical report, BRICS report RS-03-4, 2003.

[3] C. Castelfranchi. Trust mediation in knowledge management and sharing. In *Proceedings of the Second International Conference on Trust Management (iTrust 2004)*, pages 304–318, Oxford, UK, 2004.

[4] R. Demolombe. Reasoning about trust: a formal logical framework. In *2nd international conference on trust management (iTrust'04)*, pages 291–303, 2004.

[5] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *Symposium on Access Control Models and Technologies*, 2004.

[6] C. English, W. Wagealla, P. Nixon, and S. Terzis. Trusting collaboration in global computing systems. In *First International Conference on Trust Management (iTrust 2003)*, pages 136–149, 2003.

[7] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[8] T. Grandison and M. Sloman. Specifying and analysing trust for internet applications. In *Proceedings of the Second IFIP Conference on e-Commerce, e-Business and e-Government*, 2002.

[9] T. Grandison and M. Sloman. Trust management tools for internet applications. In *First International Conference on Trust Management (iTrust 2003)*, pages 91–107, 2003.

[10] A. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. The kgp model of agency. In *European Conference on Artificial Intelligence (ECAI'04)*, 2004.

[11] F. Sadri and F. Toni. A logic-based approach to reasoning with beliefs about trust. In *Workshop on Automated Reasoning for Security Protocols Analysis (ARSPA'04), affiliated to IJCAR'04*, Cork, Ireland, July 2004.

[12] S. Terzis, W. Wagealla, C. English, and P. Nixon. Trust lifecycle in a global computing environment. In *Global Computing 2004*, LNCS 3267, 2004.

[13] M. Winslett. An introduction to trust negotiation. In *First International Conference on Trust Management (iTrust 2003)*, pages 275–283, 2003.