



**HAL**  
open science

**Trust and uncertainty in distributed environments :  
application to the management of data and data sources  
quality in M2M (Machine to Machine) systems.**

Mondi Ravi

► **To cite this version:**

Mondi Ravi. Trust and uncertainty in distributed environments: application to the management of data and data sources quality in M2M (Machine to Machine) systems.. Other [cs.OH]. Université Grenoble Alpes, 2016. English. NNT : 2016GREAM090 . tel-01679344

**HAL Id: tel-01679344**

**<https://theses.hal.science/tel-01679344>**

Submitted on 9 Jan 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## THÈSE

Pour obtenir le grade de

**DOCTEUR DE L'UNIVERSITÉ GRENOBLE  
ALPES**

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

**Mondi RAVI**

Thèse dirigée par **Pr. Yves DEMAZEAU** et  
codirigée par **Dr. Fano RAMPARANY**

préparée au sein du **Laboratoire Informatique de  
Grenoble** dans l'**Ecole Doctorale Mathématiques,  
Sciences et Technologies de l'Information,  
Informatique**

**Confiance et incertitude  
dans les environnements  
distribués - application à  
la gestion des données et  
de la qualité des sources  
de données dans les  
systèmes M2M (Machine  
to Machine)**





# Contents

<b>Abstract</b>	<b>11</b>
<b>Résumé</b>	<b>13</b>
<b>Acknowledgment</b>	<b>15</b>
<b>1 Synopsis</b>	<b>17</b>
<b>2 Introduction</b>	<b>23</b>
2.1 Problem statement . . . . .	25
2.2 Motivation . . . . .	26
2.3 Our hypothesis . . . . .	27
2.4 Use cases . . . . .	28
2.4.1 Intelligent Community . . . . .	28
2.4.2 Smart City Garbage Collection . . . . .	29
2.5 Success criteria . . . . .	31
2.6 Thesis outline . . . . .	31
<b>3 State-of-the-Art</b>	<b>33</b>
3.1 Trust . . . . .	33
3.1.1 Modeling trust . . . . .	34
3.1.2 Classification based on success criteria . . . . .	42
3.1.3 Desired aspects of Trust for the IoT . . . . .	43
3.2 Uncertainty . . . . .	44
3.2.1 Sources, types and the need for managing uncertainty . . . . .	44
3.2.2 Modeling uncertainty . . . . .	47
3.2.3 Propagation of uncertainty . . . . .	50
3.3 Trust and uncertainty management in Distributed Systems . . . . .	50
3.3.1 Crowdsourcing and Data Fusion . . . . .	52
3.4 Truth Maintenance Systems . . . . .	53
3.4.1 Assumption-based Truth Maintenance Systems . . . . .	54
3.4.2 Distributed ATMS . . . . .	56
3.5 Classification of the State-of-the-Art . . . . .	57

<b>4</b>	<b>Solution approach</b>	<b>59</b>
4.1	Envisaging a distributed system as a MAS . . . . .	59
4.2	Knowledge base . . . . .	61
4.2.1	Representation . . . . .	62
4.2.2	Rules and Reasoning . . . . .	62
4.3	Modeling trust . . . . .	63
4.3.1	Eigen Trust . . . . .	64
4.3.2	Local and Global trust . . . . .	64
4.3.3	Predefined trust . . . . .	65
4.3.4	Drawbacks of Eigen trust . . . . .	65
4.3.5	$\beta$ -reputation model . . . . .	65
4.4	Modeling uncertainty . . . . .	66
4.4.1	Possibilistic Logic . . . . .	66
4.4.2	Probability Theory . . . . .	68
4.5	Agent beliefs and their relation with trust . . . . .	69
4.5.1	Interaction . . . . .	69
4.5.2	Beliefs . . . . .	70
4.6	Using trust measure as uncertainty . . . . .	70
4.7	Assumption-based Truth Maintenance System . . . . .	71
4.8	Reasoning with uncertainty . . . . .	72
4.9	Discussion . . . . .	74
<b>5</b>	<b>Implementation</b>	<b>77</b>
5.1	Distribution aspect . . . . .	77
5.2	Data representation . . . . .	78
5.3	The agent . . . . .	81
5.3.1	The agent behaviors . . . . .	83
5.3.2	The reasoner . . . . .	84
5.3.3	The rule base . . . . .	85
5.3.4	The ATMS . . . . .	85
5.4	A working example . . . . .	87
<b>6</b>	<b>Applications</b>	<b>93</b>
6.1	Intelligent Community . . . . .	93
6.1.1	Simulation setup . . . . .	94
6.1.2	Tests . . . . .	94
6.1.3	Experimentation . . . . .	96
6.1.4	Discussion on the trust and the uncertainty models . . . . .	101
6.2	Smart City Garbage Collection . . . . .	101
6.2.1	Analysis of the Data set . . . . .	101
6.2.2	Dissimilarities of the data set to our initial use case . . . . .	103
6.2.3	Objectives of this application . . . . .	103
6.2.4	Enhancements and assumptions in the data set . . . . .	104
6.2.5	Simplified trust model . . . . .	105

6.2.6	Uncertainty model . . . . .	105
6.2.7	Experimentation . . . . .	106
6.2.8	Verification of the use case . . . . .	108
6.3	Project FIWARE . . . . .	109
6.3.1	Introduction . . . . .	110
6.3.2	Home mood Scenario . . . . .	110
6.3.3	Approach and architecture . . . . .	111
6.3.4	Scenario implementation and experimentation . . . . .	113
<b>7</b>	<b>Conclusion and Discussion</b>	<b>117</b>
7.1	Contributions and Achievements . . . . .	117
7.2	Limitations and future enhancements . . . . .	119
7.3	Extending beyond the explained applications . . . . .	119



# List of Figures

2.1	Problem statement . . . . .	25
2.2	Intelligent community use case. . . . .	28
2.3	A smart garbage collection system . . . . .	30
3.1	Example: How a conclusion can be derived . . . . .	54
3.2	Example illustrating an ATMS . . . . .	56
4.1	Architecture of an agent showing different components, knowl- edge bases and the data flow . . . . .	60
4.2	An example ontology of Intelligent Community use case . . . .	62
4.3	Detailed figure of a reasoner and its relationship with the belief base . . . . .	63
4.4	A transaction between querying agent $Q$ and replying agent $R$	70
4.5	Reasoner and ATMS . . . . .	72
4.6	Dependency graph snapshots from the ATMS depicting possi- ble cases in reasoning trust relations. $A_1$ , $A_2$ are assumption nodes while $Z$ is a derived node. The corresponding labels of the nodes are shown alongside. . . . .	74
5.1	Example from the Intelligent community use case. . . . .	78
5.2	JADE facilitating communication amongst agents in two con- tainers. . . . .	79
5.3	TUM ontology. . . . .	82
5.4	ATMS class diagram. . . . .	88
6.1	Evolution of trust on two sources $B$ and $C$ , as computed by $A$ , and uncertainty of the decision taken for the corresponding iterations . . . . .	96
6.2	Memory usage in a single container . . . . .	100
6.3	The Smart city garbage collection data model (Source: A2I Systems) . . . . .	102
6.4	Some remarks and the conclusions that can be derived from them . . . . .	107
6.5	FLOD IoT Semantic Context Broker . . . . .	112





# List of Tables

3.1	Classification of trust models . . . . .	45
3.2	Classification of uncertainty models . . . . .	51
3.3	Classification . . . . .	58
4.1	Transaction table for an agent $A$ . . . . .	64
4.2	A sample remarks and reasoned information table . . . . .	69
4.3	Analysis of derivation of conclusion . . . . .	75
5.1	The initial and after communication with $M1$ states of the belief base and the ATMS of home $A$ . . . . .	91
5.2	The state of the belief base and the ATMS of home $A$ after communication with the neighbor $B$ for weather and soil information. . . . .	92
6.1	Experimentation trust update . . . . .	95
6.2	Statistics after 100 iterations . . . . .	97
6.3	Statistics after 100 iterations for almost 50% accuracy of all sources . . . . .	98
6.4	A snippet of <code>DriverHistory</code> table . . . . .	104
6.5	Trust computation based on number of remarks . . . . .	105
6.6	Remarks and their counts . . . . .	108
6.7	Remarks and different inferences . . . . .	108



# Abstract

Trust and uncertainty are two important aspects of many distributed systems. For example, multiple sources of information can be available for the same type of information. This poses the problem to select the best source that can produce the most certain information and to resolve incoherence amongst the available information. Managing trust and uncertainty together forms a complex problem and through this thesis we develop a solution to this. Trust and uncertainty have an intrinsic relationship. Trust is primarily related to sources of information while uncertainty is a characteristic of the information itself. In the absence of trust and uncertainty measures, a system generally suffers from problems like incoherence and uncertainty. To improve on this, we hypothesize that the sources with higher trust levels will produce more certain information than those with lower trust values. We then use the trust measures of the information sources to quantify uncertainty in the information and thereby infer high level conclusions with greater certainty.

A general trend in the modern distributed systems is to embed reasoning capabilities in the end devices to make them smart and autonomous. We model these end devices as agents of a Multi Agent System (MAS). Major sources of beliefs for such agents are external information sources that can possess varying trust levels. Moreover, the incoming information and beliefs are associated with a degree of uncertainty. Hence, the agents face two-fold problems of managing trust on sources and presence of uncertainty in the information. We illustrate this with three application domains: (i) The intelligent community, (ii) Smart city garbage collection, and (iii) FIWARE : a European project about the Future Internet that motivated the research on this topic. Our solution to the problem involves modeling the devices (or entities) of these domains as intelligent agents that comprise a trust management module, an inference engine and a belief revision system. We show that this set of components can help agents to manage trust on the other sources and quantify uncertainty in the information and then use this to infer more certain high level conclusions. We finally assess our approach using simulated and real data pertaining to the different application domains.



# Résumé

La confiance et l'incertitude sont deux aspects importants des systèmes distribués. Par exemple, de multiples sources d'information peuvent fournir le même type d'information. Cela pose le problème de sélectionner la source la plus fiable et de résoudre l'incohérence dans l'information disponible. Gérer de front la confiance et l'incertitude constitue un problème complexe et nous développons dans cette thèse, une solution pour y répondre. La confiance et l'incertitude sont intrinsèquement liés. La confiance concerne principalement les sources d'information alors que l'incertitude est une caractéristique de l'information elle-même. En l'absence de mesures de confiance et d'incertitude, un système doit généralement faire face des problèmes tels que l'incohérence et l'incertitude. Pour aborder ce point, nous émettons l'hypothèse que les sources dont les niveaux de confiance sont élevés produiront de l'information plus fiable que les sources dont les niveaux de confiance sont inférieurs. Nous utilisons ensuite les mesures de confiance des sources pour quantifier l'incertitude dans l'information et ainsi obtenir des conclusions de plus haut niveau avec plus de certitude.

Une tendance générale dans les systèmes distribués modernes consiste à intégrer des capacités de raisonnement dans les composants pour les rendre intelligents et autonomes. Nous modélisons ces composants comme des agents d'un système multi-agents (SMA). Les principales sources d'information de ces agents sont les autres agents, et ces derniers peuvent posséder des niveaux de confiance différents. De plus, l'information entrante et les croyances qui en découlent sont associées à un degré d'incertitude. Par conséquent, les agents sont confrontés à un double problème: celui de la gestion de la confiance sur les sources et celui de la présence de l'incertitude dans l'information. Nous illustrons cela avec trois domaines d'application: (i) la communauté intelligente, (ii) la collecte des déchets dans une ville intelligente, et (iii) les facilitateurs pour les systèmes de l'internet du futur (FIWARE - le projet européen n° 285248, qui a motivé la recherche sur nos travaux). La solution que nous proposons consiste à modéliser les composants de ces domaines comme des agents intelligents qui incluent un module de gestion de la confiance, un moteur d'inférence et un système de révision des croyances. Nous montrons que cet ensemble d'éléments peut aider les agents à gérer la confiance aux autres sources, à quantifier l'incertitude dans

l'information et l'utiliser pour aboutir à certaines conclusions de plus haut niveau. Nous évaluons finalement notre approche en utilisant des données à la fois simulées et réelles relatives aux différents domaines d'application.

# Acknowledgment

There are a lot of people that I need to thank for having helped me in different ways during this thesis. Without their help and support I would never have finished my dissertation.

First and foremost are my advisors: Dr. Fano Ramparany and Pr. Yves Demazeau. I would like to express my deepest gratitude to both of them. Being my industrial advisor, Dr. Fano was really close to me. He guided and helped me greatly with my research and writing of this thesis. His motivation, patience, valuable suggestions, and his moral support is highly appreciable. Pr. Yves, too helped me with his excellent guidance, care, patience and precious remarks throughout the thesis. I feel lucky to have found such wonderful researchers as my thesis advisors.

Orange Labs and the team “Smart Home and Access (SMA/COSY)” is made up of smart, dedicated and helpful people. My thanks to them for providing an exceptional and friendly environment for the research and development. The interactions with different people and groups in Orange Labs has led to the development of the use cases presented in the thesis. My sincere thanks to everybody involved in these fruitful discussions. The Odense Renovations Selskab (Odense Waste Collection Company, Denmark) and Alireza Derakhshan, in particular, need a special word of thanks, who agreed to share and explain the valuable data about the Odense city garbage collection with us.

My lunch buddies: Michael, Kévin, Aimé, Ludovic, Willy, Julian, Koutheir, Ding, Corentin and the members of the group\_10 team have helped me learn the French culture and the language. The diverse discussions that we have had over the several lunches have broadened my knowledge about the world. A sincere thanks to you.

Lastly a big thanks to my family. They have provided me great support and motivation during my thesis. Sri Rama Rao and Srimati Annapurna (my mom and dad), Srimati Simhachalam (my grandma), The Koppala’s (Sunita, Pinky, Souma, Krisna) - my sister’s family and my brother Ravindra, have always been a great source of inspiration. An addition to this list is my fiancé (to-be) Ms. Geetha Pandiri, who, for the past year has been pampering me with great love and care. Thank you Geethu.

My apologies to those that I may have missed. Thanks to all of you.





# Chapter 1

## Synopsis

A major problem of Distributed Systems today is having a good Quality of Information (QoI). It is a characteristic of data. It comprises several aspects of data such as accuracy, timeliness, trustworthiness, uncertainty etc. With the advent of new technologies in computing and communication such as the Internet of Things, the domain of Distributed System has grown tremendously. In this regard, our work focuses on improving the QoI by managing two important aspects of a distributed system: trust on the sources of information and uncertainty in the information. We illustrate the problems of management of trust and uncertainty further with three real world use cases. The first use case is called *Intelligent Community*. It explores a futuristic scenario where different homes in a neighborhood share information about the weather and the soil conditions to benefit mutually and operate their automated garden watering systems. The problem in this use case lies in identifying trustworthy neighboring sources and decision to take is when to turn on/off the gardening system. The second use case is called *Smart City Garbage Collection*. It shows how a city administration can profit from the crowd as possible information source along with the deployed sensor network. The crowd represents multiple sources with varying levels of trust. The decision to send a team to a particular place for garbage collection, is determined from inputs of the different sources. The third one is the development of a generic enabler to manage trust and uncertainty for a wide variety of applications. It permits the reuse of various trust and uncertainty management algorithms that we developed in this work. We evaluate our solution with respect to four key criteria: *validity of the hypothesis*, *genericity* and *applicability* of the algorithms for more than one domain, *simplicity* in terms of computations, *robustness* for large scale application domain for different data sets of the domains.

In order to understand the problems of management trust and uncertainty in distributed system, we need to understand how they function. Information exchanges are a common phenomenon in the distributed systems.

All the more so in today's world, where most of the devices are connected to the Internet. Any device or person can act as an information source and they can be heterogeneous. Upon receiving information from various sources, an entity takes up the difficult challenge of reasoning about the information. This process can be complex involving multiple levels of derivations, and finally arriving at one or more conclusions. The entity, thus confronts with the problem of selecting the best conclusion based on the available information. Even with a single conclusion, it is unable to tell how certain it is. A natural way to eliminate this problem would be to use trust and uncertainty information. In our work, we propose to model trust for various information sources, and use this as a measure of uncertainty in the information. We use this further with uncertainty logics such as possibilistic and probabilistic to propagate uncertainty towards the derived information. This enables us to choose decisions that are more certain.

Trust is a subjective term and varies widely according to the domain of interest. A general perception of trust is that someone or something is trustworthy if one can delegate a task to him/her and that the task will be completed with satisfaction. We examine trust management from a distributed system point of view, where entities interact amongst each other for the exchange of information. Trust for an entity in such systems, is generally computed as a cumulative value of satisfaction for the entity by others in the system in the past and the present. In order to learn the performance of such trust management algorithms and their suitability to the domain of IoT, we studied a few important ones in the literature. We found two of them namely: (i) Eigen Trust and (ii)  $\beta$ -reputation systems suitable in terms of scalability and simplicity for our domain. Hence, they were our preferred choices for experimentation. Of the two systems, we found the former to be scalable in domains that have distributed trust management components, while the latter is simple for systems having centralized trust management.

Uncertainty is a major cause of poor data quality. It arises from several reasons: imprecise data, unreliable and untrustworthy sources, absence of data, reasoning by abduction etc. Like trust, it has also been widely researched. Some important theories of uncertainty calculus include Probability theory, Possibility theory and Dempster Shafer Theory. In our model for uncertainty, we used Possibility theory as the data in our domain of application is generally incomplete and associating probability measures may not be possible. Dempster-Shafer, on the other hand has counter intuitive results in certain cases. We found that the use of possibilistic logic is good except for certain limitations such as computation of uncertainty for cases where conclusions are supported by multiple sets of assumptions. Hence, we have tried to study probabilistic model and Dempster-Shafer theory as an alternative. Using trust measure for reasoning about uncertainty of information is a relatively new domain of research. As these fields are themselves

quite vast, most of the researches have focused on to one of the two problem domains. This motivated us to pursue with our thesis.

Distributed systems have evolved greatly over the years, from classical client server architecture to peer-to-peer (P2P) to grid, cloud and ubiquitous computing. The Internet of Things (IoT) is a relatively new technology that facilitates distributed computing amongst devices that are heterogeneous, deployed far-and-wide and intelligent. As explained above, trust and uncertainty problems pose a big challenge for the *entities* or devices of such systems.

All information that an entity considers true are called *beliefs*. A core necessity for modeling an entity of the distributed system is proper management of the beliefs based on information from different sources. It must be able to maintain a consistent set of beliefs for the entity. When a new information arrives, it must add to its knowledge-base, derive any new information, remove the ones that become inconsistent. In other words, it must be able to maintain a dependency graph of how a belief was developed. This is called *Belief Revision System*. Of the various types of such systems available, we found Assumption-based Truth Maintenance System (ATMS) most suited for our work as it can handle managing multiple contexts at the same time. We classify the state-of-the-art along three major aspects of the thesis: Trust, Uncertainty and Distribution and along two application domains, Internet of Things and Internet of People. We find the need for more research in the joint fields of trust and uncertainty and in the modern domains of application such as the IoT.

Our approach involves modeling distributed systems as in the use cases, as a Multi-Agent System. Each of the entity of this system is then an *agent*. The information exchanges are modeled as the messages exchanged amongst the agents. We add behavior to the agents describing how they should handle the incoming messages, manage trust of the sources of information, how new conclusions can be derived and what model to use to compute uncertainty of the information. Thus, we model an agent to be consisting of the following distinct components: (1) A Trust management module, (2) An uncertainty management module, (3) An ATMS, (4) Incoming message handler, and (5) A rule engine and a rule base.

The main idea of our solution is how we transform trust into uncertainty. In most cases, data uncertainty does not accompany the data from sources. We need to mine it out of the data, which requires off-line data processing to compute the QoI values. This is undesirable for modern distributed systems like our domain of application (e.g., Internet of Things, Internet of People), as they can change rapidly and systems need to take decisions based on them. Hence, for such systems, we propose to use a distributed trust management system, that maintains trust levels of different sources of information. We then obtain data uncertainty from trust of the information source using

our hypothesis that: “In the absence of any other information, the data uncertainty is inversely proportional to the trust on the data source that furnished it”. Having computed the uncertainty for the input data, we propagate these values towards the derived information using uncertainty models, to obtain uncertainty associated with them. Thus, at the end of the reasoning and uncertainty propagation phase, we have conclusions and how strongly are they supported by the different data sources. Our approach thus makes two important improvements to the QoI of such systems: (i) Able to tell how an information is derived from various sources (ii) Quantify the uncertainty of different derivations. We used Eigen Trust and  $\beta$ -Trust models for our experimentation as they fit our requirements of simplicity of algorithms in terms of computation and scalability. Similarly, we used Possibilistic logic for uncertainty propagation for Intelligent community use case as other methods such as probability theory do not capture completely the ignorance in the information. We, later used probabilistic method to improve over drawbacks to possibilistic approach. The ATMS provides us a mechanism to explore how an information can be derived and our trust and uncertainty computation algorithm enables us to quantify uncertainty in the information. This integration of ATMS along with our trust and uncertainty management modules forms the core of our solution.

For implementation, we use Java Agent DEvelopment framework (JADE) to instantiate a group of agents. This is a stable, open source and widely known framework for Multi-Agent Systems. It supports a number of *Foundation for Intelligent Physical Agents* (FIPA) communication protocols and is highly modular and customizable for the needs of a research. In order to remain simple, we chose JADE as our distribution framework. We used linked open data in Resource Description Framework (RDF) format to represent the use cases. We developed an ontology for the domain and we assume each of the agents share this same ontology. However, their instances differ based upon the evolution of their knowledge bases. The domain specific rules that the agents use for derivation of new inferences are modeled in Semantic Web Rules Language (SWRL), as it is compatible with RDF. The information exchange is based on FIPA query interaction protocol. The agents seeking information do a query with a `query-ref` message, seeking a specific information. The responding agent either accepts and sends the `inform-result` or rejects the request. As we represent the data in terms of RDF, we use SPARQL CONSTRUCT query string as content in the query message. The `inform-result` message contains an RDF. There are other advanced MAS models based on JADE, such as Jadex BDI Agent System and BDI4Jade, but they specifically target modeling agents based on *Belief-Desire-Intention* (BDI) architecture. BDI can be an interesting choice for future extension of this work.

We evaluated our approach against three important success criteria; (1)

Validity of the hypothesis, (2) Genericity, (3) Simplicity, (4) Scalability, and (5) Distribution of algorithms. We used two models for trust; the first based on  $\beta$ -Reputation and the second based on EigenTrust, and Possibilistic logic for uncertainty. Our first experimentation involved simulated data for the Intelligent Community use case and our second experimentation involved real-world data of city garbage collection. The results show that managing trust and using it as a measure of uncertainty in the information helps devices make better decisions. FIWARE is a European project under the umbrella of the Future Internet Public Private Partnership (FI-PPP) initiative. There are a number of generic enablers (GE) defined in FIWARE. A GE offers some generic purpose functions via well-defined APIs. We exposed our approach as one such enabler called TUM (Trust and Uncertainty Management) that provides other applications to instantiate their system by declaring the sources of information, the domain specific rules and the models of trust and uncertainty to be used, and then achieve better QoI by managing trust and uncertainty.

In conclusion, we studied and illustrated the challenges of incomplete, vague, and uncertain information provided by different sources. Modeling and quantification of the uncertainty in the information is utmost important in order to distinguish between data quality of different data. From the experimentation and evaluation, we can say that the trust measure of a source can be used for quantifying uncertainty in the information. We formalized distributed systems as MAS and we modeled an individual agent and proposed various components needed by it to manage trust for other agents in the network and then reason about uncertainty in the derived or inferred information.

The Eigen-trust model that we used suffers from some limitations that need to be addressed in future models. The model does not take into account the interactions relative to time i.e. it treats all interactions with equal importance. This may not be preferable in cases where we need to detect good sources turning to bad. Also, this model does not distinguish between two sources: one that never interacted and the other that had only negative interactions because of the normalization. The possibilistic logic used in our first model too has limitations. For a conclusion supported by multiple environments with different possibility or necessity values we do not have a distinct solution. We have explored application domains related to the Internet of Things (Intelligent Community) and the Internet of People (Smart City Garbage Collection), and as a future work we wish to explore the approach being applied to more domains.



## Chapter 2

# Introduction

Information exchanges are a common part of the distributed systems. More so in today's world, where most of the devices are connected to the Internet. The exchanges are mainly related to either seeking of information from a source or acting as a information source and providing information to a seeking party. There is also a growing interest in making the devices 'Smart' i.e., make them able to reason about the context by seeking information from different sources. A smart device is an electronic device, generally connected to other devices or networks via different protocols such as Bluetooth, NFC, WiFi, 3G, etc., that can operate to some extent interactively and autonomously. With this kind of setup, where smart devices are connected to several possible sources of information, it is clear that the devices can often run into situations where they need to seek information from multiple sources. There are two major issues that they confront to at this juncture.

1. How to choose between two or more sources for seeking an information?
2. How certain is the information provided by a source?

These questions, precisely form the problem statement of our work. It relates to finding a solution for a distributed system where a peer (or an agent) needs to take its decision based on inputs from different sources.

There are two possible answers for the first question. First, the device can seek information from all sources, and then do a majority voting on the received responses to obtain the possible correct information. Second, the device can maintain trust levels on sources based on past similar experiences and pick sources with the highest trust levels for the interaction, on the grounds that trustworthy sources are better than the others. The problem with the first approach is that it is computationally infeasible in cases where there are many sources. Sending many messages means consuming a



large chunk of resources of the distributed system for processing and synchronization of messages. Also, majority consensus cannot always lead to a correct information. So, we hypothesize that trust management module can help devices in decision making. Our first objective is to explore different trust algorithms in distributed systems and compare them for specific domain such as the Internet of Things.

The second question is related to the information certainty or more generally what is known as “Quality of Information”. Information provided by the sources are generally marred with uncertainty of different types: imprecision, vagueness, incompleteness etc. Information certainty quantifies the closeness of the information to the true value. The absence of this means that one cannot guarantee the truthfulness of an information. This forms the second objective of our work where we try to model information uncertainty using several modeling techniques.

Having demarcated *trust* and *uncertainty*, the main question for us now is: “*How is the trust on the source related to the uncertainty in the information provided by it?*”. In order to answer this, we explore the relationship between trust on information sources and information uncertainty. Our hypothesis is that the trust on information source can be a measure of information certainty. Higher the level of trust on the information source, higher is the certainty of the information provided by the source. Though, information uncertainty can be due to several factors, for our work, we limit this to be based upon the trustworthiness of the source.

A summary of the objectives of our work are:

1. To explore different trust management algorithms in distributed systems and compare them for specific domain such as the Internet of Things.
2. To model information uncertainty using several modeling techniques.
3. To explore the relationship between trust on information sources and information uncertainty.
4. To develop an integrated framework that uses trust as a measure of uncertainty, and then is able to infer new information with certainty using a model of uncertainty propagation.

So far, we introduced a global view of our work. The rest of the chapter is divided into the following sections. In section 2.1, we introduce the precise problems that we tackle in this work. Sections 2.2 and 2.3 present our motivations for this work and the hypothesis that we try to prove. We illustrate the problems of the thesis with the two use cases in 2.4 and then reuse the algorithms in a third application to create a generic framework for different projects. We, then explain the success criteria that we try to study

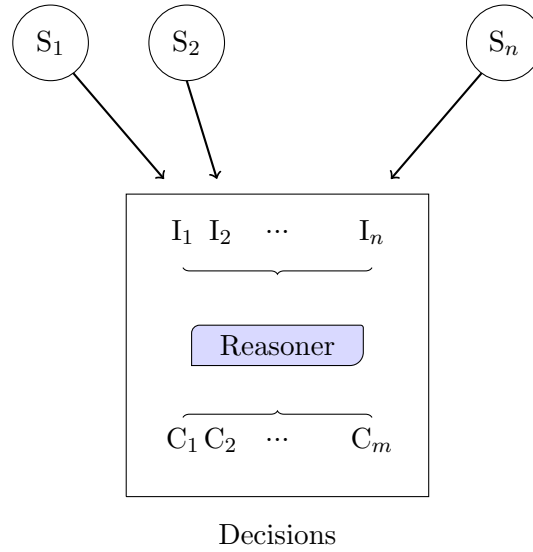


Figure 2.1: Problem statement

with our approach in section 2.5. Finally, we sketch the thesis outline in the section 2.6.

## 2.1 Problem statement

Our work deals with tackling the problems of management of trust and uncertainty in a distributed environment. To clarify the problems, we use the Future Internet that comprises two major domains: the Internet of Things (the interconnection of devices) and the Internet of People (the interconnection of the devices as well as people). The end point or the entity in such domains are the devices. The Future Internet is envisioned as a world where all devices are interconnected, accessible and manageable from wherever in the world and also the devices are assumed to be *Smart* i.e. they can reason and take decisions based on inputs from different sources. A general figure depicting this scenario is shown in 2.1. The rectangular box represents a smart device. There are  $n$  sources of information ( $S_1, S_2, \dots, S_n$ ) that may be providing both homogeneous and/or heterogeneous types of information. We call the information provided by these sources as  $I_1, I_2, \dots, I_n$  respectively. Based on these different information, the device must be capable of derive a set of conclusions represented as  $C_1, C_2, \dots, C_m$  respectively. In this regard, a device faces the following major problems:

1. How to model the trust of the information sources in different domains and how is the trust related to the uncertainty in the information? Can

- quantification of the information with a degree of certainty help the device resolve which decision to take?
2. How to choose amongst the different available information sources? At any instance of time, a device can have a number of information sources to select from. The sources can belong to different organizations and can be located in diverse locations. An ideal scenario would be to gather information from all the sources before reasoning and making a decision, but for large number of sources, that may not be feasible, both in terms of resource utilization and time constraints.
  3. How certain is an information provided by a source? Many a time, when only a piece of information is available, nothing can be told about its certainty, unless specifically supported by additional information. How can an entity obtain this information on certainty of an information provided by another source?
  4. How to infer a set of conclusions from the available information and classify them so as to select the best decision out of many that may be possible? If more than one conclusion can be derived from the available information and only one needs to be opted, the device must have criteria to select one.
  5. How to deal with inconsistent information from the sources and contradictory conclusions? In many cases, multiple sources of information implies inconsistent information and conclusions. This can have an adverse impact on the decision making process. Selecting one or the other conclusion without supporting evidence can lead to an undesirable result.

Having pointed out the problems faced by a distributed system in the absence of trust and uncertainty management mechanism, we present the motivations for undertaking this thesis in the following section.

## 2.2 Motivation

The study of uncertainty and trust have attracted the attention of several researchers all over the world with regards to different domains of application. We are no exception. For us, there are two major motivations for this thesis.

1. Theoretical motivation: As per our research, there has not been an extensive study of trust and uncertainty together. The theoretical motivation for this work can be explained in the following three points.

- (a) The growth of distributed systems and the Quality of Information (QoI): In the past couple of decades, the field of distributed systems has seen an immense growth. From millions of devices in 2000 to over 16 billion connected devices today<sup>1</sup>, we have come a long way with the help of scalable protocols and the notion of connecting everything to the Internet. But, a fundamental question one needs to answer is : “Has the growth contributed to a good QoI that exists on the Internet?”. With data uncertainty present in most information systems the answer has to be *no*.
  - (b) Lack of mechanisms in to manage trust of sources and quantify uncertainty in information: Trust and Uncertainty are two domains that have been extensively studied. However, very few have targeted to find a relationship between the two. It would be interesting to quantify uncertainty in information based on the trust of the information source. As per our knowledge, such mechanism has not be studied together for distributed system like the Internet of Things. So, this was one big motivation to pursue with this work.
  - (c) Embedding intelligence in devices: The enhancements in distributed systems have led to extend the centralized intelligent systems to spread out the intelligence to the end points or devices. Such systems generally use rule-based engines, reasoning with *if-this-then-that* type rules.
2. Applicative motivation: One of the projects that the funding of this thesis is attached to is the project - FIWARE. It is a collection of projects dealing with the Future Internet. The researchers in the various sub projects within FIWARE studied the impacts of uncertainty in distributed systems and contemplated having some uncertainty management framework for its several sub-projects. This gave birth to the idea of managing uncertainty with the help of sources of the information. The project FIWARE is explained in detail in section 6.3.

In the following section we present our hypothesis to solve the problems that we introduced in the section 2.1.

## 2.3 Our hypothesis

In order to quantify the uncertainty in the information provided by a source, we assume that we can have a mechanism in the device or in the system that stores, provides and updates the trust levels of the information sources for that device. Our hypothesis is that the sources with higher trust levels

---

<sup>1</sup><http://newsroom.cisco.com/feature-content?type=webcontent&articleId=1208342>

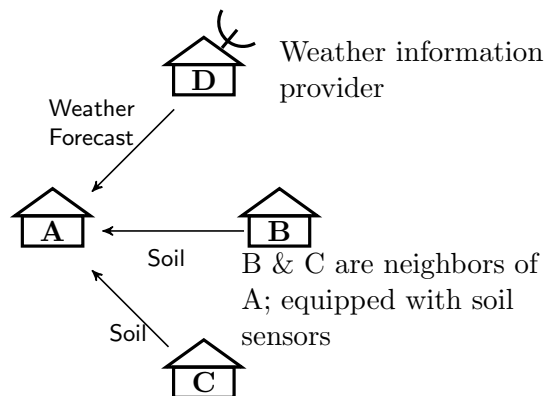


Figure 2.2: Intelligent community use case.

will produce more reliable information than those with lower trust values. Mathematically, at any point of time, the uncertainty in the information provided by a source of information is inversely proportional to the trust of the source. We further assume that the uncertainty of the information thus obtained, can be used to infer high level conclusions supplemented with uncertainty, and that such conclusions are better than those derived without uncertainty.

## 2.4 Use cases

In this chapter we introduce the use cases that illustrate the problems of management of trust and uncertainty in distributed domains. The first use case described in section 2.4.1 presents a Smart Home based futuristic application where a community of houses fitted with smart devices communicate among themselves to manage their garden watering. The second use case described in section 2.4.2, deals with the smart city garbage collection, where inputs from citizens and sensors are used together to reliably predict the presence of garbage at different places of a city. Apart from these use cases, we have an extension of this work as a generic enabler for its reuse in a group of other projects. We explain this in section 6.3.

### 2.4.1 Intelligent Community

The use case illustrated in figure 2.2, considers three homes  $A$ ,  $B$  and  $C$  in a neighborhood of a community and a weather forecast station  $D$ . Homes  $A$ ,  $B$  and  $C$  are smart, meaning that they are equipped with devices connected to the Internet and thereby able to benefit from the services of different information providers. A typical example of such a service is obtaining weather information from weather forecast companies. Depending upon the

information, the devices can alert the users, or take the decision to activate an actuator to control the garden watering system. In this use case, we assume that home *A* is not equipped with any specific sensor and thereby has to rely only on the weather forecast from *D* (in particular the chance of rain) to control the garden watering system.

Homes *B* and *C*, that are the neighbors of *A*, on the other hand have sensors such as soil moisture detector deployed for their use. Furthermore, the owners of homes *B* and *C* are open to sharing soil moisture and humidity readings with their neighbors. We try to explain how home *A* can leverage this additional information source to significantly improve its decision-making ability about when to water his/her garden. Since, the homes are located in close vicinity of each other, we assume that their soil and weather conditions do not vary much and thus for home *A*, the neighboring homes serve more or less as trustworthy sources of information.

#### 2.4.2 Smart City Garbage Collection

Smart city is a modern concept where information sharing (from a crowd of people, sensors or things) along with the existing physical infrastructure of the city enables a sustainable economic development in the city, leading to a well managed natural resources and high quality of life [CDBN11]. Some examples include Smart energy management, Smart public transportation system, Smart noise and pollution control etc. We consider one such smart city application - “Smart Garbage Collection”, to explain the need of trust and uncertainty management.

Usually an agency or a company is responsible for gathering all the garbage from the city and for disposing it to an appropriate location. The agency plans a static itinerary (daily, weekly) to send its vehicles and personnel to each and every corner of the city and collect the garbage. To make this process smarter the agency makes use of “intelligent containers”<sup>2</sup>. These containers have a specially designed wireless sensor that detects the amount of garbage in them and relay the information in real-time to one of data collector, which in turn relays it to the central information system. Thus, these containers form a wireless sensor network. The information obtained from this network allows the agency to plan a better itinerary and to improve its efficiency for garbage collection. However, the sensor network cannot be made omnipresent. Perhaps, because of the cost constraints or because garbage may be thrown randomly on the streets. In some cases, the citizen may be more careless and just let the garbage on the road, not even close to the container.

In order to deal with the problem of omnipresence of the sensors, the agency makes use of crowd-sourcing mechanism such as telephone calls from

---

<sup>2</sup><http://www.urbiotica.com/>

citizens, or a web-portal to report the random garbage and/or garbage near the garbage collection point. Thus, the problem of smart garbage collection can be further aided by crowd-sourcing.

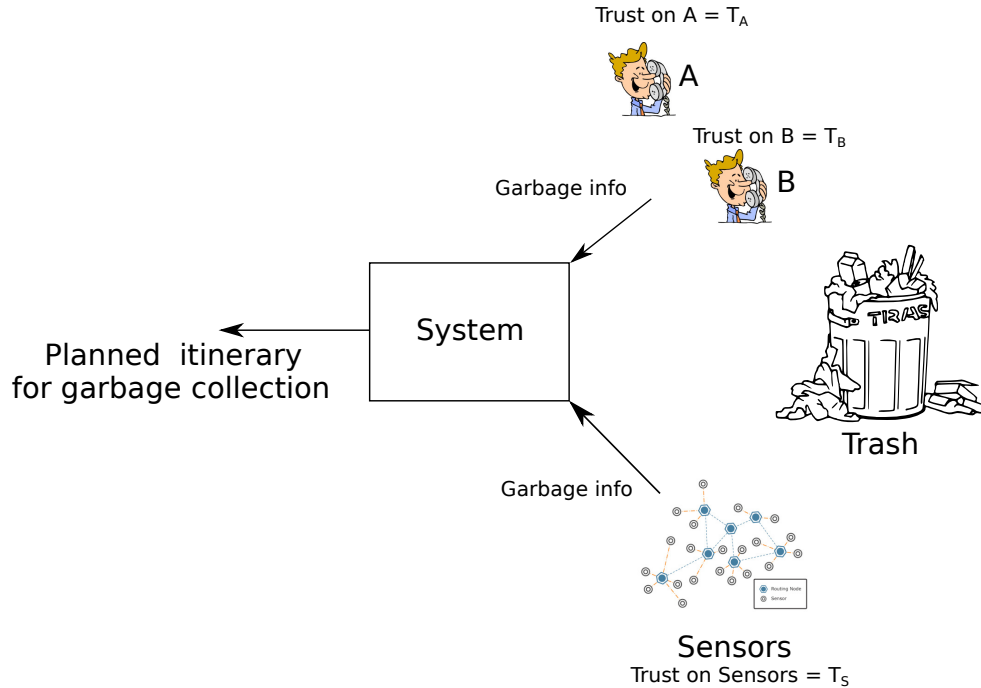


Figure 2.3: A smart garbage collection system

Figure 2.3 shows a block diagram of such a system. Citizens  $A$  and  $B$  call the system and inform about garbage lying near their neighborhood. Additionally, there may be some information from the deployed sensors  $S$  in the streets. Based on the information received from the sources, the system then instructs the garbage men to get the garbage collected from that location.

The problem with such a setting is that there can be more than one source reporting an information, there is always an uncertainty on which information source to rely on to take a decision. Another issue is how does the number of sources which report the same information affect the trust of the agent on the information. We seek to find the solutions to these problems in this work. So far in the chapter, we presented the problems of management of trust and uncertainty, the motivation of our work, our hypothesis and the use cases illustrating the problems. In the following section, we describe the success criteria on the basis of which we assess our solution.

## 2.5 Success criteria

In this chapter, we have so far presented the problems pertaining to lack of a proper mechanism to manage of trust of the sources and uncertainty in the information with the help of the use cases. We also introduced the motivation for this work and the proposed hypothesis to resolve the problems. We now present a list of success criteria that will help us assess our solution and algorithms.

1. **Validity of the hypothesis:** We proposed a hypothesis that trust of the source of information and the information certainty follow a proportional relationship. This success criterion is to check whether the hypothesis is true or not with respect to different application domains and different data sets.
2. **Genericity:** Since, we intend to apply the solution to more than one problem domain: one relating to the *Internet of Things* and the other related to *Internet of People*, this criteria states that the proposed trust model should be generic enough to be easily extended to newer domains of application.
3. **Simplicity:** The application domains we introduced in 2.4 are time-critical. Any trust and uncertainty management algorithm that is computationally time-consuming is not preferable. They should be simple enough so that they do not affect the performance of the system as the number of entities in the domain increases. Hence, with this success criteria, we intend to measure the performance of the trust and uncertainty models when the number of entities in the system goes up.
4. **Scalability:** Our goal in this thesis is to resolve problems relating to incoherent data or data sources. The model should be reliable enough to be applicable to a situation involving incoherent data or data sources at large scale. With this success criterion, we intend to measure the performance of the overall solution when the number of entities in the system goes up.
5. **Distribution:** This is a key aspect of the Internet of Things. With this success criterion, we aim at verifying how the algorithms for the management of trust and uncertainty can be distributed to the entities of a distributed system.

## 2.6 Thesis outline

The remainder of this thesis is organized as follows:



- In Chapter 3, we present the state-of-the-art in the domains of trust and uncertainty. We explore different models of trust and uncertainty that can be applicable to the application domains. We classify the interesting works with similar problem set based on whether they relate to the Internet of Things or the Internet of People.
- We then explain our approach in Chapter 4. It includes envisaging the application domains as Multi Agent Systems and incorporating several components like the reasoner, the ATMS, the trust and uncertainty models together to form smart agents that have ability to reason.
- We present the implementation of our work in Chapter 5. Here, we discuss the details of how we realized the different components of an agent and explain its working with an example.
- In chapter 6, we revisit the use cases and explain the different tests and experimentations that we carried out to achieve our success criteria. We also present the details of the Trust and Uncertainty Management (TUM) module created as our contribution towards the project FI-WARE.
- We finally conclude with Chapter 7, where we present our conclusions and discussions.

## Chapter 3

# State-of-the-Art

Trust and Uncertainty have been widely studied research topics. The problems of management of trust and uncertainty have been associated with respect to different domains of applications. Hence, there already exists a huge literature and a plethora of models for trust and uncertainty. Our goal in this thesis is to find a relation between trust and uncertainty, how trust values of a source can be used as a measure of uncertainty in the information provided by the source and how an entity of a Distributed System can reason over it. Here, we present the state-of-the-art in the domains, that we found interesting for distributed systems, in particular the Internet of Things.

In section 3.1, we begin by exploring the works that exist in the domain of Trust and examine them from the perspective of applying to the Internet of Things (IoT) and the Internet of People (IoP) domains. We then do the classification of different trust models that we found interesting in section 3.1.2 and we present some aspects that we found necessary for modeling trust for IoT and IoP in 3.1.3. Next, we present a survey in the domain of uncertainty, its modeling and using its propagation in section 3.2. In 3.3, we present some recent works that aim to manage uncertainty and trust from crowdsourcing domain specifically. Belief revision is an important aspect of modeling an autonomous/semi-autonomous entity such as an agent of a Multi Agent System. In order to maintain a consistent set of beliefs and generating explanations for conclusions, we study a specific class of belief revision systems called the Assumption-based Truth Maintenance Systems in section 3.4. We finally do a classification of all works that we found interesting with respect to the domains of IoT and IoP in section 3.5.

### 3.1 Trust

Modern distributed systems consist of entities (homogeneous or heterogeneous) working in tandem, exchanging information to reason and take decisions. As the entities may belong to different organizations, or because they

may be connected to sensors with different precision levels, the information provided by them are not reliable to the same degree. Hence, it is utmost important for the entities to maintain a quantitative value of trust for other sources it has interacted with, depending upon the ratio of good to bad interactions.

Quantitatively, trust is a value that someone or something bestows upon other(s), for the accomplishment of a certain task. It can be imagined to be composed of different components each reflecting different aspect as explained in the subsection 3.1.1. The list is not exhaustive, as we limit ourselves to components that we think are useful for modeling the things of the M2M systems.

### 3.1.1 Modeling trust

Like described earlier, Trust is multifaceted. It is interpreted differently in different domains. As such, we explain here the different models that we found in the literature. Broadly, we classify the models into three categories: (i) Computational models (ii) Socio-cognitive models and (iii) Security based models. These models are explained in detail in the section 3.1.1. In section 3.1.2, we compare these models in terms of our success criteria. Based on this study, we then present different aspects of trust that are most important from our applications perspective in section 3.1.3.

#### Computational models

These aspects enable easy quantitative representation of trust. Reputation, recommendation, ratings etc. are some of the manifestations of such aspects. Most models of computational trust assume the trust to be within a numerical range say  $[0,1)$ , and then normalize the value of trust of each entity in the system relatively. Many models use *reputation* as a measure of trust. Reputation is defined as a collective measure of trust put upon an entity by the whole society. Generally, there is a centralized authority that collects, stores and manages reputation of different entities in a system (e.g. online marketplaces such as eBay, Amazon, StackOverflow). However, there are also systems that need a distributed mechanism to store and manage reputation (for instance, in Peer-to-Peer systems that lack a central authority). From this point of view, trust can be seen of two distinct categories: *Local* and *Global* trust. For distributed systems, an important aspect to consider is how the trust is computed: either based on the experiences of an entity that needs to compute the trust itself or all entities of the system.

- *Local Trust*: A local value of trust is one that is computed by the involved parties for each other. A local trust value may or may not be shared. Since, the local trust is obtained from personal experience of an entity, it can rely on it completely. However, to have a local

reputation value for all entities of the system, an entity needs to have interactions with all other entities.

- *Global Trust*: A global value is an accumulation of all the transactions that have ever occurred. All agents are free to query about the global value of trust related to any other agent, we assume that the agents will share this information. A global reputation value is available to all, even if the querying agent is a newcomer to the society. It reflects the global view about an agent, hence an agent's personal view may be overridden by the global trust.

Below, we present some important computational trust models.

1. **Marsh's model**: Marsh's work [Mar94] is one of the first in proposing computational trust model. He presents trust as a continuous variable over a specific range  $[-1, +1]$   $+1$  being blind trust (very high level of trust) and  $-1$  representing complete distrust. The work presents an agent's (trustee) trust on another agent (truster) to be composed of following components:
  - Basic trust: It is a representation of the general trusting disposition of an agent. It signifies whether the agent is optimist, pessimist or realist. It is obtained from all the past interactions of the agent ranging within  $[-1, +1]$ .
  - General trust: Given two agents,  $x, y$ ;  $T_x(y)$  denotes the trust agent 'x' has on agent 'y'. It is within the range  $[-1, +1]$ . It is not relative to any specific situation.
  - Situational trust: Denoted as  $T_x(y, \alpha)$ , this represents the trust the agent 'x' has on agent 'y' specific to a situation 'alpha'. It takes value within the interval  $[-1, +1]$ .
  - Importance: It is an agent-centered or subjective judgment of a situation on the part of the agent concerned. It is represented as  $I_x(\alpha)$  and has values within the interval  $[0, +1]$ .
  - It also models other trust components such as *utility* of a delegated task to the agent, the *risk* involved in accomplishing the task and *cost/benefit ratio*. Further, the model also tries to model an agent's optimistic or pessimistic behavior in trust modeling.

However Marsh's model does not present trust from the distribution perspective i.e., how to compute and resolve trust in a distributed environment. Hence, we searched for a trust model that fits distribution aspect. Eigen trust is one such model that can be applied to distributed domain. We present this in the following section.

2. **Rahman's model** [ARH97]: Rahman's trust model presents trust from a distributed environment perspective, where one may need to compute trust about other peers asking for recommendations for the known peers. Three main points are put forward in this work: (i) A decentralized approach to trust management of information sources in a network, (ii) Generalization of trust beyond management of keys and authentication (iii) A protocol for the exchange of trust information. This model permits the computation of trust value of a source that can be reached through a series of recommendations. In other words, it models trust transitivity. A real world scenario can have any path of recommendation from an entity to the other. Rahman's model is a good solution in such cases.
3. **Eigen Trust** [KSGM03]: Eigen Trust is a trust model that uses the ratings of different interactions amongst the nodes to compute a quantitative value of trust. The goal of this work is to identify the sources of inauthentic files and bias users against downloading them. The method they employ is to give a *trust value* to a peer based on its previous behavior. It distinguishes between:
  - (a) Local trust: The opinion that peer  $Q$  has of peer  $R$  based on experiences based on its own experiences with  $R$ . Any interaction between  $Q$  and  $R$  is given a value +1 or -1 depending on whether it was good or bad. If  $Tr(Q, R)$  represents each of such interactions then the local trust is given by:

$$s_{QR} = \sum Tr(Q, R) \quad (3.1)$$

This sum can have a positive or a negative or a zero value. Since, a negative value for trust does not make sense, the authors propose to normalize this value so that it is within the range [0,1]. The normalized local trust is obtained as follows.

$$c_{QR} = \frac{\max(s_{QR}, 0)}{\sum_k \max(s_{Qk}, 0)} \quad (3.2)$$

The normalization ensures that the sum of the trust values of all the peers is equal to 1, i.e.,  $\sum_{i=1}^{i=n} c_{iR} = 1$ .

- (b) Global trust ( $t_i$ ): The trust that the entire system places on peer  $i$  is the global trust of  $i$ . In order to obtain this, each peer asks all its neighbors to provide their local trust information about every other peer. This value of trust allows overcoming the problem of limited experience.

If  $A$ ,  $B$  and  $C$  are three peers in a system and  $t_A$ ,  $t_B$  and  $t_C$  are the global values of trust on them respectively, then newer values

can be obtained from the various local trust as follows:

$$t_A^{new} = t_B * c_{BA} + t_C * c_{CA} \quad (3.3)$$

$$t_B^{new} = t_A * c_{AB} + t_C * c_{CB} \quad (3.4)$$

$$t_C^{new} = t_B * c_{BC} + t_A * c_{AC} \quad (3.5)$$

where  $c_{ij}$ ,  $i$  and  $j \in A, B, C$  are the local trust values of  $i$  on  $j$  and  $t_A^{new}$ ,  $t_B^{new}$ ,  $t_C^{new}$  are the new global trust values. In general, this can be rearranged in the form of a matrix as shown in equation 3.6 below. Here, we assume that there are  $n$  peers in the system. The Eigen vector of the matrix  $C^T$  gives the global trust values of the different peers, and hence the trust model is called Eigen Trust.

$$\begin{pmatrix} t_1^{new} \\ \dots \\ t_k^{new} \\ \dots \\ t_n^{new} \end{pmatrix} = \begin{pmatrix} c_{11} & \dots & c_{k1} & \dots & c_{n1} \\ \dots & \dots & \dots & \dots & \dots \\ c_{1k} & \dots & c_{kk} & \dots & c_{nk} \\ \dots & \dots & \dots & \dots & \dots \\ c_{1n} & \dots & c_{kn} & \dots & c_{nn} \end{pmatrix} \begin{pmatrix} t_1 \\ \dots \\ t_k \\ \dots \\ t_n \end{pmatrix}$$

or,

$$t_i^{new} = C^T t_i \quad (3.6)$$

If the peer seeks opinions the second time

$$\vec{t}_i = (C^T)^2 \vec{c}_i \quad (3.7)$$

and continues in this fashion for  $n$  times then,

$$\vec{t}_i = (C^T)^n \vec{c}_i \quad (3.8)$$

If  $n$  is sufficiently large, this will converge to the same vector for every peer  $i$ , yielding the Eigen trust values. The details of the whole computation is presented in algorithm 1.

**Further enhancements to Eigen Trust** The algorithm presented above in 1 has three major problems.

- (a) Inactive peers or new ones' join the system. So, such peers have small or no values of trust.
- (b) A group of peers in the system act malicious. The model presented does not have mechanism to scale down or nullify its effect.
- (c) The convergence of trust values in algorithm 1 may take a long time.

In order to deal with these problems, Eigen Trust introduces the notion of Pre-Trusted peers. These may reflect the real world scenario

**Algorithm 1:** Basic EigenTrust

---

**input** : 1.  $\vec{e}$ : An zero vector representing the initial trust of all the nodes,  
 2.  $C^T$ : The transpose of the matrix of local trust of all nodes as shown in equation 3.6  
 3.  $\epsilon$ : A threshold value below which the trust vector is assumed to have converged.  
**output**:  $\vec{t}$ : A vector representing the global trust of all the nodes in the system

```

1       $\vec{t}^{(0)} = \vec{e}$ 
2      repeat
3           $t^{(k+1)} = C^T \vec{t}^{(k)}$ 
4           $\delta = ||t^{(k+1)} - t^{(k)}||$ 
5      until  $\delta < \epsilon$ 

```

---

where we know the reliability of the peers beforehand. If  $P$  is the set of peers which are known to be pre-trusted, then, a function can be defined that provides equal pre-trust weights to each of the pre-trusted peers as:

$$p_i = \begin{cases} 1/|P| & \text{if } i \in P \\ 0 & \text{otherwise} \end{cases}$$

and for the new and inactive peers

$$c_{ij} = \begin{cases} \frac{\max(s_{ij}, 0)}{\sum_k \max(s_{ik}, 0)} 1/|P| & \text{if } \sum_k \max(s_{ik}, 0) \neq 0 \\ p_j & \text{otherwise} \end{cases}$$

To counter the malicious peers and clusters, the authors introduce a constant  $a$  which is less than 1.

$$\vec{t}^{(k+1)} = (1 - a)C^T \vec{t}^{(k)} + a\vec{p} \quad (3.9)$$

The modified algorithm is presented in 2. As shown, the algorithm uses a predefined vector to initialize trust of all the peers and the trust update step in the loop uses the formula of equation 3.9. In other words, the new values of trust of different peers obtained from this method is a weighted mean of the trust computed using Eigen trust explained before and the pre-trusted values assigned to the peers.

Though the Eigen trust is shown to be robust and scalable, it suffers from some drawbacks. First, the trust values of the peers are normalized, i.e., the values are relative to each other. This does not provide

---

**Algorithm 2:** Basic EigenTrust using predefined trust values
 

---

**input** : 1.  $\vec{p}$ : A pre-defined vector representing the initial trust of all the nodes,  
 2.  $C^T$ : The transpose of the matrix of local trust of all nodes as shown in equation 3.6  
 3.  $\epsilon$ : A threshold value below which the trust vector is assumed to have converged.  
**output**:  $\vec{t}$ : A vector representing the global trust of all the nodes in the system

```

1          $\vec{t}^{(0)} = \vec{p}$ 
2         repeat
3              $t^{(k+1)} = C^T \vec{t}^{(k)}$ 
4              $t^{(k+1)} = (1 - a)C^T \vec{t}^{(k+1)} + a\vec{p}$ 
5              $\delta = ||t^{(k+1)} - t^{(k)}||$ 
6         until  $\delta < \epsilon$ 

```

---

an absolute interpretation when needed in certain domains. Second, the negative transactions are all considered as zero trust. So, we cannot distinguish between two peers: one which has no transactions at all and the other with all the negative transactions. And finally, Eigen trust proposes to eliminate contribution of malicious peers by introducing predefined trust vector for the peers. This may not be realistic, as trusted peers may become untrustworthy over time. In the following section, we present another reputation based trust model that can alleviate the problems of the Eigen Trust.

4. **The  $\beta$ -Reputation System** [JI02] The  $\beta$ -Reputation system is based on using the beta probability density functions to combine feedback and derive reputation values. The simplicity of the reputation model makes it attractive for research on our domain of interest (IoT) where scalability is of immense importance. According to the authors, posterior probabilities of binary events can be represented as beta distribution. The beta distribution is indexed by two parameters  $\alpha$  and  $\beta$  and can be expressed using the gamma function as

$$f(p|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha) \Gamma(\beta)} p^{\alpha-1} (1-p)^{\beta-1}, \text{ where } 0 \leq p \leq 1, \alpha > 0, \beta > 0 \quad (3.10)$$

For this the probability value  $p \neq 0$ , if  $\alpha < 1$  and  $p \neq 1$ , if  $\beta < 1$ . The expectation of this beta distribution function is

$$E(p) = \frac{\alpha}{\alpha + \beta} \quad (3.11)$$



If a binary event  $x$ ,  $\bar{x}$  is given and if  $r$  and  $s$  are their respective number of occurrences, then probability density function of the occurrence of the event  $x$  based on the past observations is given by

$$\alpha = r + 1 \text{ and } \beta = s + 1 \text{ where } r, s \geq 0 \quad (3.12)$$

Combining the above equations, the expectation value of the reputation function can be expressed as:

$$E(p|r, s) = \frac{r + 1}{r + s + 2} \quad (3.13)$$

The authors argue that the binary events can be used to represent the positive and negative feedbacks for an interaction between two entities. If  $X$  and  $T$  are two entities, then  $r_T^X$  and  $s_T^X$  represent the number of positive and negative feedbacks provided by  $X$  about  $T$ , then the normalized trust value is called the *Reputation rating* and is obtained as:

$$Rep(r_T^X, s_T^X) = (E(f(p|r_T^X, s_T^X) - 0.5) \cdot 2) = \frac{r_T^X - s_T^X}{r_T^X + s_T^X + 2} \quad (3.14)$$

Like Eigen trust, this model too suggests the need for normalization of trust values. The range that the authors propose is  $[-1, +1]$ . Equation 3.14 is simple yet strong as it can model reputation in terms of basic mathematic operations. Further, the authors also provide methods to combine reputations from two different entities and discounting the reputations based on beliefs.  $\beta$ -Reputation can serve as a good alternative for modeling trust. Though the authors do not explain the trust computation from a decentralized perspective, they argue that it can be updated without much change.

5. **CertainLogic** [RHMV11]: The paper proposes a model for the evaluation of propositional logic terms under uncertainty that is compliant with the standard probabilistic approach and subjective logic. It presents a method to calculate 1. the trustworthiness of the subsystems and atomic components 2. the uncertainty associated to this information. The authors call the logic to represent trust and uncertainty as CertainLogic (for evaluating propositional logic subjected to uncertainty). They provide definition for CertainTrust (CT), Expectation value of CT, Operators OR, AND, NOT to calculate opinions about truth of the given propositions, Commutativity and Associativity. CT is defined to be composed of three parameters (i) average ratings, (ii) certainty and (iii) initial expectation value. Thus, the model is useful where the participating entities are related by logical operations.

6. **Behavioral trust** In [GW11], Gligor and Wing try to differentiate between computational and behavioral trust. They say that computational trust defines trust relations among devices, computers and networks while behavioral trust is the trust among the people and organizations. Hence, for computational trust, the authors state that the underlying communication model must be secure and possess mechanism to verify the authenticity (correctness and trustworthiness) of the end users or devices and fault-tolerance to recover from any harm caused by malicious sources. For humans, the trust is much more than computational trust. They present behavioral trust to model trust in humans. They view this as a one-shot game between the trustor and the trustee that includes a punishment for bad behavior. Though, the work presents important points for management of trust, the implementation aspect is unclear.
7. **Peer-2-Peer Trust** Aberer et al. [AD01] present a trust model that permits to manage and store the reputation of peers in a distributed manner. The core of this work is the P-Grid: a distributed data storage [Abe01]. It provides an infrastructure for efficient storage and search of information in a peer to peer manner. The trust model proposed by Aberer et al. is based on the behavioral information of the different peers. It combines the complaints about a peer by other peers for obtaining a value of trust of the peer. The authors suggest that the trust thus modeled is highly scalable. However, it seems complicated to be tested.

Thus, in this section, we presented some of the important computational models that we found in our research. In the following subsections, we present socio-cognitive and security based trust models, which are important from the Internet of People type of application domain.

### Socio-Cognitive models

These aspects include modeling behaviors such as: competence, willingness, persistence, motivation of an entity for computing the trust values of the agents. The seminal work of Castelfranchi and Falcone [CF01] is of great importance in this respect. They present trust as composed of specific beliefs and goals, with special attention to evaluations and expectations. The specific beliefs are related to the competence, willingness, motivation, benevolence of the agent to which a task is to be delegated to. Moreover, they say trust on an agent is with regards to a goal and a context. The model is extensive in the sense that it also includes the external conditions such as dangers, opportunities, obstacles etc. that can arise due to the delegation of a task. This model presents trust from several social and behavioral perspectives. However, modeling of such aspects in a real implementation

is questionable. Also, modeling of such behaviors for domains involving non-humans such as the IoT may be inappropriate.

A well known paradigm in socio-cognitive models is *Belief-Desire-Intention* (BDI) based models [GPP<sup>+</sup>98]. A BDI agent reasons based on its mental attitudes: beliefs, desires and intentions. *Beliefs* represent the facts about the world including the inference rules that allow obtaining new information. *Desires* are the motivational state of the agent or the objectives of the agent. Intentions are the desires that the agent is committed to as an execution plan. Various reasoning systems such as [DLG<sup>+</sup>04], [GI90] are based on this. The major drawbacks of such models are: (i) lack of ability to learn from the past behaviors, (ii) lack of mechanisms to be applied to for interactions, in particular to Multi Agent Systems. (iii) lack of explicit goal representation.

### Security based models

Security is an important aspect of modern information systems. It is measured in terms of the underlying infrastructure of the systems to cope to encrypting information to guarantee properties such as authentication, privacy, confidentiality, integrity and non repudiation. Such aspects include the securing end to end communication channel between the entities under communication. The better the communication channels, higher is trust on the system.

In [Jøs96], Jøsang presents trust from information security perspective. He gives real world examples about what trust is for humans and how humans reason based on it. He uses the same notion to represent trust for a rational agent in a distributed system. The most essential component of trust for him in the paper is the knowledge about the world. Trust of a source can be assessed based on it. In [HG12], the authors present trust and uncertainty from a mobile application user interface design perspective. They discuss the design requirements of the applications. The explanations they provide are very high level from the application point of view and a user's interaction with the application.

Since, we have mostly tried to cover important works in trust from the Internet of Things perspective, we may have missed some literature. Hence, we point out to some interesting surveys in trust that discuss it at length and with regards to different application domains. Some of them are [RHJ04], [VRAC10], and [HHRM12].

### 3.1.2 Classification based on success criteria

In the previous subsection, we described several trust models. Here, we classify these models based on the various success criteria that we discussed in section 2.5 and our desired application domains. The classification is

shown in the table 3.1. As seen in the table, Eigen trust meets most of our success criteria.  $\beta$ -reputation model and P2P trust also fit well.

### 3.1.3 Desired aspects of Trust for the IoT

As we presented in the earlier section, trust is multi faceted and there is a vast literature in modeling trust for different domains of applications. However, the IoT being relatively new innovation, trust for the IoT has not been studied in depth. In [LS12], the authors present an overview of different models of trust that may be interesting for the IoT and they include some of the works that we described in the earlier sections. Hence, with the goals of modeling trust for the IoT, we present here the aspects taken from our research, that we think are important.

1. Reputation: With the capability of things or devices of the IoT possessing the capability of storing experiences its interactions with others, reputation as a trust measure seems to be the most fitting aspect. Storing all the transactions an entity has with others allows it to compute a numerical value of reputation (as in Eigen trust or  $\beta$ -reputation). However, in systems with frequent sharing of information, the transactions can be large in number, and storing all of them may pose a problem. In this regards, a policy to identify the importance of a transaction may be needed, and then the entity can store the important transactions only. A second issue to think about is whether the reputation computed locally by an entity should be allowed to share with the entire system or not or how should the system manage the transactions of the entire system. In works like P2P Trust, Eigen Trust etc., the authors suggest that trust can be stored in a distributed manner, meaning that no one entity stores its own transactions but they are stored by the system somewhere in the network.
2. Security: This is another important aspect that can determine how trustworthy a system is. This is particularly important in domains that include entities belonging to different organizations. Various mechanisms such as Public Key Infrastructure, secure data transfer channels are key to build a secure system. This aspect is particularly interesting in domains like cloud computing where services are offered by different service providers. Trust a service provider over other may mean how secure the underlying infrastructure is compared to others.
3. Timeliness: Trust does not just depend on the quality of reply of the responding entity but it depends also on when the response arrived. Generally, a requesting entity waits for a certain period before which it considers all arriving responses as futile even though they may be of good quality. Thus, timeliness of data is an important factor. How

much should the window of time be before considering the arriving data as useless depends on the application domain. Nonetheless, a source that provides information in a timely fashion is preferable.

## 3.2 Uncertainty

Like trust, uncertainty is a widely researched domain. Quantitatively, uncertainty is the parameter that measures the dispersion of a range of measured values. Modern information systems produce a large amount of data. But, many times, the data does not indicate how certain it is. There can be several reasons as to why an information or the underlying data is uncertain. Vagueness, unclear statements, untrustworthy sources etc. are some examples. Knowing uncertainty about an information can guide us to take better actions and conclusions. In this work, we seek a way to quantify uncertainty in information and explore how it can be used to derive new information with a degree of certainty.

In this regard, we present a review of the works in the domain of uncertainty. We begin by looking at the works that provide an insight to what are the reasons for uncertainty and what are its different classes in subsection 3.2.1. Then, we present the well known modeling techniques to model uncertainty in section 3.2.2. Propagation of uncertainty for inferring new information is our interest in particular. So, we present how different models of uncertainty can be used to achieve that in section 3.2.3.

### 3.2.1 Sources, types and the need for managing uncertainty

[MBF<sup>+</sup>12] is a white paper on sources and management of uncertainty. It was published in conjunction with a European project FIWARE. The project envisages the use of the Future Internet in a variety of areas such as Transport, Smart Cities, Agrifood etc. The researchers found that data uncertainty to be a big challenge and hence the need for proper management mechanism across all domains. It explains the problems and sources of data uncertainty, also called, the Quality of Information (QoI). The authors point out nine specific factors that degrade QoI in various domains. They are listed as follows:

1. Trustworthiness/credibility/reliability of data source: There is a need to quantify the trustworthiness of different data sources. E.g., Trust of various users that report information is not the same.
2. Accuracy of sensors: The sensors may or may not provide accurate information for the domain because of various reasons like the sensors are of poor quality, sensors are faulty etc. The accuracy of the information may not be sufficient for the domain of application.

Table 3.1: Classification of trust models

Model	Genericity	Simplicity	Distribution	Scalability	Pros	Cons
Rahman's model	✓	+	✓		Trust transitivity	
Marsh's model	✓	+			Comprehensive, generic	Unclear about distribution and computationally complex
Eigen Trust	✓	++	✓	✓	Simple, Distributed, Relative measure	No notion of negative trust
$\beta$ -reputation	✓	+++			Simple, Based on statistical background	
CertainLogic	✓	+				
GligorWing Model	✓				Models both: trust for devices and humans	Not an explicit model as such
P2P Trust	✓	+	✓	✓	Explicit distribution, good implementation	Complex to test
Socio-cognitive models					Models mental attitude, applicable to model human trust	Difficult to model for devices
Security based models	✓					

3. Timeliness of information: There may be a time lag between an occurrence of event and its storage in the actual system.
4. Uncertainty of data introduced due to the result of an abduction and reasoning process: The available information gives an indication of certain observation, but it is not a definitive conclusion.
5. Absence of data: Non-responding sensors or sensors not deployed in the area of interest leads to the absence of data.
6. Discrepancy in information representation: The representation used for storing the information may not be suitable for the domain of application. E.g., the choice of 2D projection of the earth may or may not suit the domain of application.
7. Fit for purpose: Data captured for a particular purpose may or may not fit for other purposes.
8. Chain and propagation of uncertainties/noisiness of processing steps: An information may have been derived from a number of processes. At each process, additional uncertainty or noise may get added to the existing values.
9. Belief and Plausibility of measurements and information: The plausibility represents a measure of certainty for the occurrence of an event. Though, we may have information from highly reputed source of information, there may be a need to quantify the information with plausibility.

This document serves as a good motivation to find a solution for data uncertainty. The various sources of uncertainty pointed out above can broadly be grouped into three classes: (i) Imprecise measurements, (ii) Human error and (iii) Uncertainty introduced due to data propagation.

Another work that explores the reasons for uncertainty and possible ways to correct is [ABB<sup>+</sup>08]. This is a draft from a working group from Germany that includes Universität Bremen, Deutsche Laboratories, Siemens and University of Illinois. It presents a list of desirable properties of trust representations, sources of uncertainty and research challenges in the field of uncertainty and trust. The primary objective of the working group was to study the relation between trust and uncertainty in distributed reputed systems. Based upon their experiences, they list out some important characteristics on Uncertainty and trust. According to them the sources of uncertainty are as follows:

1. Uncertainty about the identity of the interaction partner.

2. Uncertainty in the behavior of the interaction partner: A partner may behave in an unintended manner such as cheating which may lead to severe problems like system crash.
3. Uncertainty in observation: Even if an interaction partner cooperates, we may observe misbehavior due to noisy “sensors” and “channels”.
4. Second-hand experiences: Such experiences not only reduce uncertainty, but also introduce additional uncertainty like: uncertainty about the reliability of an experience provider, uncertainty about the interpretation of an experience due to different system models and subjective world view, uncertainty about the transferability/applicability of an experience as the context of experience may be different, uncertainty about the temporal accuracy of an experience since the behavior of a transaction partner may change over time.

These are important notes on what are the causes of uncertainty. The group further explains the need of a strong trust framework to counter uncertainty. According to them, the desirable properties that trust representations must have are: (i) A trust representation should reflect/integrate different “uncertainties”, (ii) A trust representation should allow for decision making, and (iii) Scalar vs. complex trust representation e.g.: a trust representation must include complete explanation about the interaction context so that the context is clear.

### 3.2.2 Modeling uncertainty

There are different types of uncertainty in Distributed Systems. In this subsection, we explain them and their possible sources.

There are different ways of modeling uncertainty. In this subsection, we list some common uncertainty modeling methods and explain our choice of model for representing uncertainty.

#### Probabilistic Logic

This logic uses the theory of probability along with deductive logic to quantify how certain or uncertain a conclusion is. E.g., we take a simple example of garden watering system where the system needs to turn on the water tap only if there is no rain prediction and that the soil is dry. We can represent this with following propositional statements.

1.  $NoRain \wedge DrySoil \implies WateringON$
2.  $NoRain$
3.  $DrySoil$



Given propositions 1, 2 and 3, the deductive logic helps to *deduce* or *entail* the proposition *WateringON*. The probabilistic logic extends this further by assigning probability values to each of the propositions, and then using the laws of probability to deduce the probability related to the conclusion. This is the most widely accepted and used method to model uncertainty.

### Possibilistic Logic

A possibility distribution assigns each element  $u$  of the universe of discourse  $U$ ,  $\pi(u) \in [0, 1]$ . The uncertainty in a possibilistic clause is represented by two measures: Possibility ( $\Pi$ ) and Necessity ( $N$ ). For all  $p, q \in U$ , possibility and necessity measures satisfy the following axioms:

1.  $\Pi(\perp) = N(\perp) = 0$ ,  $\Pi(\top) = N(\top) = 1$
2.  $\Pi(p \vee q) = \max(\Pi(p), \Pi(q))$
3.  $N(p) = 1 - \Pi(\bar{p})$
4.  $N(p \wedge q) = \min(N(p), N(q))$
5.  $N(p) \leq \Pi(p)$

### Certainty Factor

The certainty factor (CF) model [Hec92] is a method for management of uncertainty and is a standard method in rule-based systems. Like probability, each event is assigned a measure of belief. The measures are usually in the range  $[-1, +1]$ ;  $+1$  indicates with utmost certainty that the event or information is true while  $-1$  indicates that the contrary is certain. The model allows assigning a belief value to the rules also. CF is computed from probabilities using two measures: (i) Measure of belief (MB) (ii) Measure of disbelief (MD). If a hypothesis  $H$  is supported by an evidence  $E$  then, these measures can be computed as,

$$MB = \begin{cases} 1, & \text{if } p(H) = 1 \\ \frac{\max(p(H|E), p(H)) - p(H)}{1 - p(H)}, & p(H) < 1 \end{cases}$$

$$MD = \begin{cases} 1, & \text{if } p(H) = 0 \\ \frac{\min(p(H|E), p(H)) - p(H)}{0 - p(H)}, & p(H) > 1 \end{cases}$$

Both these values are within the range  $[0, 1]$ . These equations are invented equations and not derived from any theory. The CF of the hypothesis  $H$  given the evidence  $E$  is then given by

$$CF(H, E) = MB - MD \quad (3.15)$$

If  $E_1$  and  $E_2$  represent two evidences with certainty factor  $CF_1$  and  $CF_2$  respectively, then the combined certainty factor of the two evidences can be obtained using the following formula.

$$CF(H, E_1 \wedge E_2) = \begin{cases} CF_1 + CF_2 (1 - CF_1), & \text{if } CF_1, CF_2 > 0 \\ CF_1 + CF_2 (1 + CF_1), & \text{if } CF_1, CF_2 < 0 \\ \frac{CF_1 + CF_2}{1 - \min(|CF_1|, |CF_2|)}, & \text{if } \text{sign}(CF_1) \neq \text{sign}(CF_2) \end{cases}$$

Certainty Factor is a simple model of uncertainty management and is fairly easy to integrate with rule based systems. However, there have been critics that argue the theoretical validity of the model. Further, the model assumes that the conditions in a rule are always independent which is not always true.

### Subjective Logic

Subjective logic is an extension of probabilistic logic which combines uncertainty represented in terms of probability with opinions about the uncertainty about a given information. A binomial opinion about truth of a proposition  $x$  represented as  $\omega_x$  is an ordered quadruple  $\omega_x = (b, d, u, a)$  where:

- $b$ : belief - is the belief that the proposition is true
- $d$ : disbelief - is the disbelief that the proposition is false
- $u$ : uncertainty about the proposition
- $a$ : base rate - this determines in the absence of any evidence about a given source, what level of trust to put in any source of the community.  $a \in [0,1]$ .

Here,  $b + d + u = 1$ , and  $b, d, u \in [0,1]$ . If  $r$  and  $s$  are the number of positive and negative evidences about the proposition  $x$ , then opinion parameters can be expressed as:

$$\begin{aligned} b &= r/(r + s + 2) \\ d &= s/(r + s + 2) \\ u &= 2/(r + s + 2) \end{aligned}$$

### Dempster-Shafer Theory

Dempster Shafer Theory [S+76] written in short as DST, is a classic way of combining evidences originating from different sources. It assigns a belief mass to each of the elements of the universe. It differs from Bayesian approach in the sense that the sum of a belief of a fact and its negation need not be equal to 1. Both can be 0, which implies that there is no evidence for

or against the fact. Having assigned the probabilities to each of the elements of the universe of discourse also called the belief masses or basic probability assignment (bpa), the theory advocates computation of two measures of a fact  $A$ : (i) Belief or  $\text{bel}(A)$ : a sum of masses of elements which are subsets of  $A$  (ii) Plausibility or  $\text{pl}(A)$ : a sum of masses of the sets that intersect with set  $A$ . Using these measure, the theory puts forward the use of a formula to combine the evidences from two or more sources to compute the combined belief.

Though, DST seems to has been widely known for data fusion from multiple sources, there are critics who argue that it can be misleading to interpret belief functions to represent “probabilities of an event” or “degree of trust on a proposition” [Pea14]. In [Zad86], Zadeh explains with examples, how DST can lead to counter-intuitive results in case of conflicts between the information provided by the sources.

### 3.2.3 Propagation of uncertainty

During the process of reasoning low level of information is used for reasoning and decision making. The reasoning process may include multiple levels and each level may involve logical operations between different information. This section explains the details of how uncertainty of the derived or inferred information can be computed from the inputs. As presented in 3.2.2, each of the models has a specific formula for deriving conclusions from conjunctions of a number of propositions. E.g., in possibility model, the possibility of a conclusion is the maximum of the possibilities of the individual propositions. We present a classification of these models in the table 3.2 from the perspective of our success criteria. We ignore distribution aspect here, as we need uncertainty propagation mechanism within an entity. By scalability of the model, we mean how the model performs as the number of propositions from which we can derive a conclusion, increases.

As seen in the classification, we find possibilistic logic is particularly interesting to us as the information exchanged is generally incomplete. This is close to our use cases explained in section 2.4. Hence, in our first approach, we studied how this can be used to model and compute uncertainties of the various conclusions.

## 3.3 Trust and uncertainty management in Distributed Systems

Trust provides a measure of the truthfulness of information source, while uncertainty is a characteristic of the information itself. In the absence of any other data, we can take the trust as an uncertainty measure. This aspect of using trust as an measure of uncertainty interests us greatly, as we think

Table 3.2: Classification of uncertainty models

Model	Genericity	Simplicity	Scalability	Pros	Cons
Probabilistic Logic	✓	++	✓	Simple	Does not model ignorance or incompleteness in information
Possibilistic Logic	✓	+++	✓	Simple min/max arithmetic based, Models ignorance in information	
Certainty Factor	✓	+	✓	Applicable to rule based systems	Lack of a proper theory. Events must be independent.
Subjective Logic	✓	++		Based on statistical background	
Dempster Shafer Theory	✓	++		Useful for data fusion	Counter-intuitive results

the different distributed domains can be enhanced to manage trust of the various sources relatively easily without much change to the system. Though there have been immense studies in modeling and resolving the problems of managing trust and uncertainty individually, their study together has not been exploited largely. However, there have been some recent work that address this. In this section we present a list of researches that focus on managing trust and uncertainty together.

### 3.3.1 Crowdsourcing and Data Fusion

One interesting domains of applications that has come up in the last decade is crowdsourcing. It is the process of accomplishing a task by assigning subtasks to a group of people or crowd, mostly an online community. Some examples are Amazon mechanical turk, Wikipedia etc. In order to obtain good results, the crowd is classified into a number of groups based on their performance and expertise. Their reports is then fused together to make the results reliable. Though, we are not directly interested in data fusion, it would be good to understand what measures of trust/reliability/uncertainty do the researchers use for the fusion. Below we present some important works that are used for crowdsourcing and data fusion.

#### Venanzi's model

Venanzi's model [VRJ13] is a recent work that addresses the problem of fusing untrustworthy reports provided from a crowd of observers, while simultaneously learning the trustworthiness of individuals. The authors use trust parameters of the sources to scale uncertainty in the estimates reported by them. They construct a likelihood model for computation of the trust of sources and propose a data fusion algorithm based on trust. They apply their work to cell tower localization in the OpenSignal project.

The use of trust measures to scale uncertainty is very relevant to our work. However, the likelihood model for trust presented by the authors does not take into account prior knowledge about user reliability. This can be a major issue, as trust is a commodity that augments from the experiences. Also, the work is presented from the point of view of a system that collects report from the users. So, it is hard to tell how it can be applied to other domains that need distribution.

#### DST based trust revision model

[SFP<sup>+</sup>13] is an important work that is based on Dempster Shafer's Theory (DST) to revise trust of different information sources. There are two major contributions from this work: (i) Proposition of *SDL-Lite*, a framework that combines Description Logic (DL) and DST to reason about uncertain infor-

mation obtained from different sources (ii) Trust revision of the information based on information from one or more information sources.

These two works ([VRJ13] and [SFP+13]) support the need for further research on trust and uncertainty together.

### 3.4 Truth Maintenance Systems

An intelligent entity needs some mechanism to maintain a consistent set of beliefs or information in its knowledge base. Along with that it must be capable of explaining how and why an information can be reasoned. *Belief Revision Systems* or *Truth Maintenance Systems* (TMSs) are component in artificial intelligence that permits to do this. They allow the management of information and its dependencies from which they may have been derived. Hence, to model an entity of distributed systems such as IoT, we think a TMS is an integral part.

There are many classes of TMSs, viz; Justification based Truth Maintenance System (JTMS), Assumption based Truth Maintenance System (ATMS), Logical based Truth Maintenance System (LTMS). ATMS is superior to other variants of TMSs in the sense that it is capable of maintaining and reasoning with a number of simultaneous and possibly incompatible sets of assumptions. Hence, we explore further the working of ATMS. We are particularly interested in how ATMS can be distributed amongst different peers of the distributed system and how can it be extended further to use uncertainty values along with maintaining consistent belief base.

#### Why ATMS?

To illustrate further the need of the ATMS, let us consider a small example that depicts a scenario in a distributed system entity. It depends on other sources of information to enrich its beliefs and take appropriate actions. Based on the inputs and beliefs, a conclusion can be deduced in one or more ways. As an example, let us consider a smart device that controls the garden watering system. It may receive information from disparate sources about the weather condition (the information about whether it will rain or not) and about the current soil conditions (whether dry or wet). Let `Dry soil` and `No rain` represent the actual values. Based on these information, the device can infer to “Switch on watering”. This is shown in the figure 3.1. It shows that “Switch on watering” can be derived by two ways. In general, this can be large in real world cases and it can also be multi-leveled. Now, in order to identify how certain a conclusion is, the device needs to know how it was derived. This demonstrates the need of a mechanism to obtain the explanations for a conclusion. Another need is the update of the existing information. In the example, if the device knows with certainty that it will

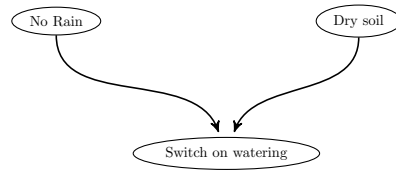


Figure 3.1: Example: How a conclusion can be derived

rain, it will need to update the derivations accordingly. These requirements support the need for an ATMS, as it provides these features.

Having described why we need an ATMS, we now present details about how it functions.

### 3.4.1 Assumption-based Truth Maintenance Systems

An Assumption-based Truth Maintenance System [DK86], also called Belief Revision System, is a system for maintaining consistent set of beliefs in the knowledge base of an agent.

#### ATMS keywords

1. Node: A node is associated with an instance of a data structure which is being manipulated by the problem solver. The actual content of datum is of no interest to ATMS. It is the problem solver that requests ATMS to create an autonomous node, thereby informing ATMS that it is reasoning with the associated data. The nodes are mainly of two types. A *premise* node that represents a truth meaning the agent believes in the information unconditionally. A second type of node is called the *assumption*. The information contained in such node are uncertain. There is another node called the *derived* node that can be obtained from one or more premise(s), assumption(s) and/or other derived node(s).
2. Justification: A justification is a statement indicating that the truth of a conjunction of nodes is sufficient to conclude the truth of a node.

$$n_1 \wedge n_2 \wedge n_3 \wedge \dots n_k \rightarrow n_x \quad (3.16)$$

where  $n_i, i \in [1, k]$  are called antecedent nodes and  $n_x$  is called the consequent node as it is derived from the conjunction of the antecedents.

3. Assumption: Some nodes as decided by the PS, are called assumptions. They are the nodes on which any datum ultimately depends. They are considered true unless proven false.

4. Environment: An environment is a set of assumptions representing the conjunction of these assumptions. If an environment  $E = a_1, a_2, a_3, \dots, a_m$  is in the label of a node  $n$ , then the ATMS has already deduced that  $a_1 \wedge a_2 \wedge a_3 \wedge \dots \wedge a_m \rightarrow n$ , i.e.,  $E \rightarrow n$
5. Label: As nodes, justifications and nogoods are added, the ATMS maintains a label for each node. A label is a set of environments, representing their disjunction, which supports the associated node. A label of a premise is an empty set  $\{\{\}\}$ . The label of an assumption node is a set containing the information itself. E.g., if an information “The weather is rainy” is represented as an assumption node `RainyWeather`, its label is  $\{\{\text{RainyWeather}\}\}$ . A derived node has one or more environments in its label. E.g.,  $\{\{\text{RainyWeather}\}, \{\text{WetSoil}\}\}$  represents the label of a node that can be derived from two environments `RainyWeather` and `WetSoil`.
6. Nogood: A nogood is a set of nodes which cannot be all true at the same time, i.e, a set of nodes derives *False*.  
 $n_1 \wedge n_2 \wedge n_3 \wedge \dots \wedge n_k \rightarrow \text{False}$ , where  $n_i$  is a node.

### Propagate and Weave Algorithms

Maintaining a consistent set of labels and constructing new labels for new nodes are main tasks of an ATMS. **Propagate** and **Weave** are the core algorithms that help perform these tasks. When a node is derived from one or more nodes like one shown in the example in 6, new environments need to be constructed to reflect the derivation of new node. The label thus created for the node  $n$  should observe four principles: (i) Minimality: No environment in the label of the node is a proper subset any other. (ii) Completeness: Every environment in the label of the node is a superset of some other environment. (iii) Soundness: The node  $n$  holds in each of the environment of the label and (iv) Consistency: All environments of the label of the node  $n$  are not nogoods. A complete description of how these algorithms work are explained in [DK86]. Below we present an example of how ATMS manages information internally, along with the different concepts that we have explained so far.

### Example

To understand the working of the ATMS, let us reconsider the example that we presented earlier in 3.4. Let us consider the information that “it will not rain” and “Soil is dry” are represented by `NoRain`



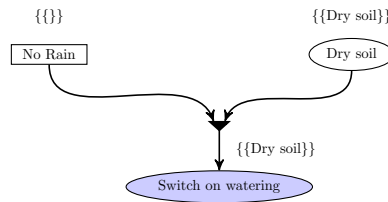


Figure 3.2: Example illustrating an ATMS

and `DrySoil`, and that we can derive another information “Switch on watering” from these information. Further more, let us suppose that `NoRain` is a premise, and `DrySoil` is an assumption. Then, these information are input to the ATMS as follows.

- (a) Declare `NoRain` as a premise and `DrySoil` as an assumption.
- (b) Declare a justification that we can derive `Switch on watering` from the two nodes declared above, i.e.,

$$\text{NoRain} \wedge \text{DrySoil} \rightarrow \text{Switch on watering} \quad (3.17)$$

The labels and the derivation of the node in the ATMS is shown in the figure 3.2. The `NoRain` node being a premise is represented by a label with empty environment and the assumption node `DrySoil` is represented by itself as an environment. The derivation of the node `Switch on watering` is done using the Weave and Propagate algorithm.

### 3.4.2 Distributed ATMS

The notion of distribution with respect to ATMS has existed for long. The work of Kraetzschmar [Kra97] on Distributed ATMS (DATMS) is interesting in this respect. He showed the use of ATMS for a distributed planning problem use case. In the work, he assumes that all agents are honest and whatever information is communicated about the assumptions are considered true. The difference here between a single ATMS and DATMS is: viewing the ATMS to be composed of many instances that communicate with each other and each of the DATMS unit maintains a difference between nodes created by itself and those that are communicated. In case of an update to the assumption related to a communicated node, the DATMS makes a callback thereby updating the assumption in the original DATMS. For this, Kraetzschmar envisioned the classic ATMS to having three different components: (i) A communication and control unit (CCU) that communicates with other DATMS peers and saves the communicated information (ii) A context management subsystem that manages context along with the

communicated knowledge (iii) A dependency network management subsystem that manages the dependency network within the DATMS. This work is interesting to this thesis as it involves distributed belief maintenance. However, viewing DATMS as a single entity of a distributed environment is debatable. An ATMS is basically a dependency network management system and cannot be seen as a reasoning and communication/management system itself. In order to be modular, it may be worthy to keep ATMS as an independent entity and use it in components of the distributed system when required.

### **3.5 Classification of the State-of-the-Art**

In the earlier sections, we have described the various important aspects of trust, uncertainty and distribution, and some important models of trust and uncertainty existing in the literature. We studied these domains from the perspective of our prospective applications concerning the Internet of Things (IoT) and the Internet of People (IoP). Hence, here we classify the three core domains of the thesis (Trust, Uncertainty and Distribution) against the domains of application. The classification is presented in the table [3.3](#).

Table 3.3: Classification

	<b>Trust</b>	<b>Uncertainty</b>	<b>Distribution</b>
IoP	[ARH97], [WS07], [CF01], [TLRJ12], [Mar94], [GW11], [CG03], [JMP06], [GYL12], [LY09], [Jøs96], [RHMV11], [HWS08], [VRJ13],	[Hal98], [PS93], [PM93], [Zad86], [RHMV11],	[MS01], [NNFM12],
IoT	[ARH97], [HG12], [WS07], [Mar94], [LS12], [GW11], [RPGH08], [CG03], [JMP06], [GYL12], [LY09], [Jøs96], [XL02], [HWS08], [VRJ13]	[Hal98], [PS93], [PM93], [Zad86], [Pla12], [CCB <sup>+</sup> 06],	[MS01], [NNFM12], [PPR11]
Others	[MBD12], [Hur06], [RHJ04] – Survey, [LS07] – Security, [LV07] – Security, [HG12] – Security, [VRAC10] – Survey	[KKRMvK09], [ABB <sup>+</sup> 08], [SKL09], [BG10], [EY09], [YC05], [MBF <sup>+</sup> 12], [WGE06],	

## Chapter 4

# Solution approach

This chapter explains our approach to solve the problems of management of trust and uncertainty in distributed domain. We envisage a distributed system as a Multi Agent System (MAS). An overall functional diagram depicting different components of intelligent agent is shown in the figure 4.1. The agents of the system store the information as propositional statements. These are a set of beliefs that the agent develops from its interactions with other agents of the system. A set of domain specific rules allow the agent to derive new inferred statements. The statements are associated with uncertainty derived from trust levels of the source of the information.

We begin the chapter by explaining the details about the representation used for the knowledge base of the agents in section 4.2. We then illustrate how we model the trust of the sources and uncertainty in the information in sections 4.3 and 4.4 respectively. In section 4.5, we present how the models of trust and uncertainty are used with agent's beliefs. Section 4.6 explains how and why we use trust measure as uncertainty in the information. Assumption-based Truth Maintenance System (ATMS) is an important component within the agent for computing the uncertainties of several derived information. We explain the details of this in section 4.7. In section 4.8, we present the reasoning process of an agent and finally we conclude the chapter by a brief discussion.

### 4.1 Envisaging a distributed system as a MAS

The goal of our work is develop a framework for a distributed system (e.g., the Internet of Things) that enables the management of trust of different information sources and quantification of uncertainty in the information derived from them. Since, trust and uncertainty handling is inherent to each entity of a distributed system, we need to model them from the perspective of each entity. A question that needs to be answered at this point is: *Is it that the entities themselves manage their trust and assess information un-*

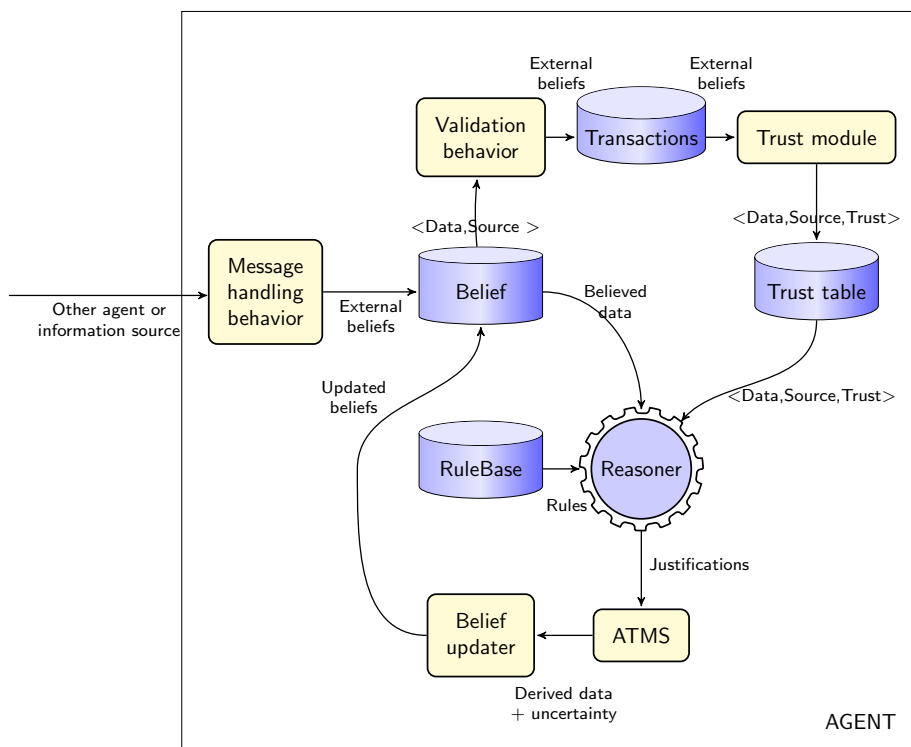


Figure 4.1: Architecture of an agent showing different components, knowledge bases and the data flow

*certainty, or is there a centralized entity to do this task specifically?* Our approach needs to be flexible enough to take this into account. Moreover, the entities can be heterogeneous and can be spread across multiple organizations. E.g., for a home in the use case 2.4.1, a weather information provider represents a source that is different from its neighbors. So, we envisage a distributed system as a Multi Agent System (MAS), where the entities can be modeled as independent agents. This has additional advantages:

1. **Simpler and scalable:** The system can be divided into agents based on the tasks they need to perform. New agents can be added easily and the solution can scale up or down when needed without much implementation changes.
2. **Parallelism:** The trust model can be distributed (e.g. Distributed Eigen Trust) in nature. With MAS, the trust can be computed simultaneously in different agents, and a global view about the trust of agents of the system can be achieved faster.

## 4.2 Knowledge base

Knowledge base is the memory of an agent. It consists the facts (true in all interpretations), beliefs (knowledge that is true to a degree of belief) and rules about the world to reasoning newer information from the existing ones. We also use the word *belief base* synonymously as the beliefs form the major part of the knowledge base of an agent. The beliefs of an agent may develop from information obtained from different sources, and that an agent may have different degrees of trust. Hence, not all information are believed equally by an agent. To model this, we associate a degree of uncertainty with information. Well-known facts and information from highly trustworthy sources have high degree of belief associated with them (In probabilistic terms, uncertainty  $\approx 0$ ) and vice versa. A belief base is, thus, a collection of beliefs of an agent at any instance of time. The source of beliefs in the belief base are mostly external information sources while some beliefs are well known facts (universal truths) or facts learnt from the past. At regular intervals of time, these beliefs in the belief base are validated. The validation procedure is performed either by comparison of the belief with true value of the information from a more trustworthy source (E.g., if a belief suggests that it will rain in the next two hours, it will be validated by available weather information in two hours) or by direct human intervention (E.g., the agent proposes the user an action and the user accepts or rejects it) or interaction with other agents. At the end of a validation process, the beliefs are converted to transactions. For practical purposes, the beliefs that have been converted into transactions can be safely disposed.

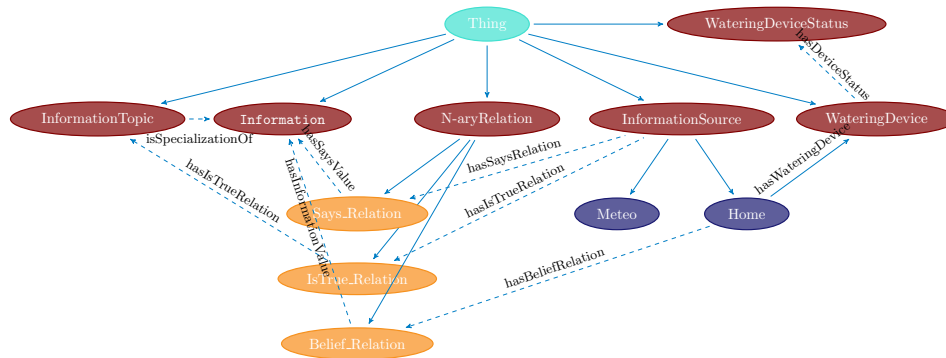


Figure 4.2: An example ontology of Intelligent Community use case

### 4.2.1 Representation

Many techniques exist when it comes to representing a knowledge base. Logical representations such as Propositional logic, Predicate logic, semantic graphs, production rules, ontology etc. We model the knowledge base as ontology. The major advantages of representing knowledge as ontology over other methods are:

- It is a well known standard and inter-operable in terms of expressing any type of information.
- It is a decentralized and a distributed format, meaning the information may be split amongst different agents and each of them understand it.
- It expresses relationship between different classes explicitly. Hence, it provides a natural way to search, explore and filter a knowledge base for related properties.
- It can be created easily from data i.e., It is data-driven.

An example ontology (Intelligent Community use case) is shown in the figure 4.2. The ellipses shown maroon and orange are subclasses of the “Thing” class. The relationships amongst different classes are shown with dashed arrows. The ellipses in blue are the instances of the class.

### 4.2.2 Rules and Reasoning

Rules are the mechanism to derive new information. A “Rule Engine” enables to infer new information from existing ones and a “Reasoner” is a component that uses a rule engine to chain two or more rules and/or information in the belief base to form a line of reasoning. A set of rules are defined and stored in the *Rule base*. Based on these rules and the believed data from the belief base, a reasoner is able to infer new information. A

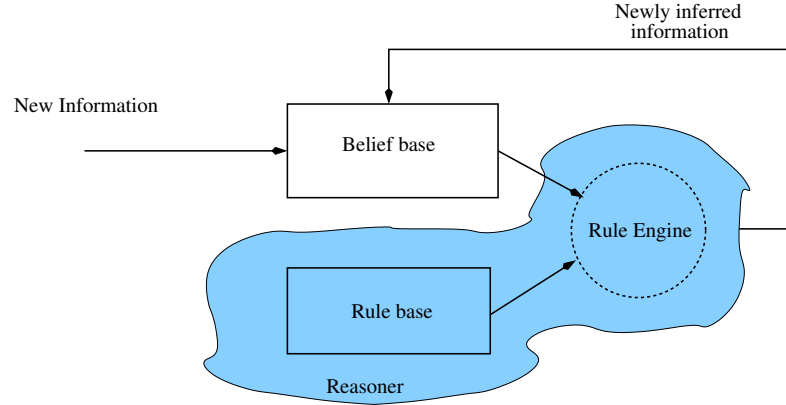


Figure 4.3: Detailed figure of a reasoner and its relationship with the belief base

detailed diagram of the reasoner and its relationship with the belief base is shown in the figure 4.3. As illustrated in the figure, the components shown in blue together form the *Reasoner* component. The reasoning process does not just infer new information, but deduces uncertainty of this inferred information.

We represent the trust on the source of information and uncertainty in the information by relationships named *hasIsTrueRelation* and *hasNecessityValue* respectively. The rules convert the trust associated with different information sources to compute the uncertainty of the inferred information based on the chosen uncertainty model (discussed in subsection 3.2.2). For possibilistic model of uncertainty, if  $X$  and  $Y$  are two clauses whose conjunction produces  $Z$  and that  $X$  and  $Y$  are given to be true with necessities  $N_1$  and  $N_2$  respectively, then from the resolution principle, we can deduce  $Z$  with a necessity value given by minimum of  $N_1$  and  $N_2$ . Thus, reasoner does not just infer the information  $Z$ , but computes the uncertainty of  $Z$ .

$$\begin{array}{c}
 X \wedge Y \rightarrow Z \\
 \text{hasNecessityValue}(X, N_1) \\
 \text{hasNecessityValue}(Y, N_2) \\
 \hline
 Z \wedge \text{hasNecessityValue}(Z, \min(N_1, N_2))
 \end{array} \tag{4.1}$$

### 4.3 Modeling trust

From our detailed state-of-the-art study about the existing trust models, we found that Eigen Trust [KSGM03] and  $\beta$ -Reputation are two models that



Table 4.1: Transaction table for an agent  $A$ 

Querying Agent (Q)	Replying Agent (R)	Service ( $\psi$ )	Rating $Tr_i(Q, R, \psi)$
$A$	$D$	WeatherInfo	+1
$A$	$B$	SoilInfo	-1
$A$	$C$	SoilInfo	+ 1

fit our requirements of being simple, scalable and applicable to distributed domains. We present the details of Eigen Trust and  $\beta$ -reputation in subsections 4.3.1 and 4.3.5.

### 4.3.1 Eigen Trust

We reuse the notion of trust computation mechanism defined by Eigen-Trust [KSGM03]. According to this work, the querying agents rate the providing agents for the service offered. The value can be +1 or -1; +1 indicates a satisfactory transaction while -1 indicates the opposite. We include service type to emphasize that trust is dependent on the type of service offered by the agent as an agent can offer many services. An illustration of the table of transactions for an agent  $A$  is shown in the Table 4.1. The example shows queries from the  $A$  to other agents  $B$ ,  $C$  and  $D$ . The first row in the table states that agent  $A$  queried  $D$  for *WeatherInfo* type of information for which  $D$  was provided a rating  $+1$ , indicating a satisfactory transaction for  $A$ . The second and the third rows are the transactions for *SoilInfo* type of information out of which the transaction with  $C$  was satisfactory and the transaction with  $B$  was unsatisfactory.

### 4.3.2 Local and Global trust

It is important to distinguish between local and global values of trust. A local value of trust is one that is computed by the involved parties for each other. A local trust value may or may not be shared. A global value is an accumulation of all the transactions that have may ever occurred. All agents are free to query about the global value of trust related to any other agent, we assume that the agents will share this information. If  $Tr_i(Q, R, \psi)$  represents  $i$ th transaction of the  $n$  transactions between a querying agent  $Q$ , a responding agent  $R$  and about a type of service or information  $\psi$ , then the local trust value is given by:

$$s_{QR,\psi} = \sum Tr_i(Q, R, \psi)$$

The normalized value of the local trust is given by:

$$c_{QR,\psi} = \frac{\max(s_{QR,\psi}, 0)}{\sum_j \max(s_{Qj,\psi}, 0)} \quad (4.2)$$

### 4.3.3 Predefined trust

An application domain generally comprises of heterogeneous sources. Some of them may be well-known, in the sense that they are owned by us, while others may be external sources. In such scenarios, it is evident that trust on the well known sources should be higher than the exterior sources. This is modeled using *Predefined trust*.

### 4.3.4 Drawbacks of Eigen trust

Eigen trust model suffers from the following important drawbacks.

1. The model does not specify a method for calculation of trustworthiness of an information when two or more sources provide the same information.
2. The model does not take into account the interactions relative to time. E.g., if a source has 3 positive interactions and 3 negative interactions; the trust model doesn't take into account when the interactions occurred.
3. The model does not distinguish between two sources which have negative interactions and no interactions at all.

Overall, Eigen trust is a scalable trust management algorithm that seems promising to be explored for the Internet of Things and M2M systems.

### 4.3.5 $\beta$ -reputation model

To overcome the limitations of Eigen Trust mentioned in the section 4.3.4, we studied the  $\beta$ -reputation model. [JBXC08] is a well-known trust model proposed by Jøsang in 2002. The model is based on Bayesian Network and beta probability function. In the pure beta reputation, users rank the transactions in a binary mode which is either satisfied or unsatisfied. However Jøsang extended the work to support float ranks that can model partly satisfaction. The satisfactory and unsatisfactory transactions are counted through two numeric variables  $r$  and  $s$  respectively. Then a statistical probability of being the next outcome satisfactory is estimated using beta density function as follow:

$$Rep_x = \frac{r_x - s_x}{r_x + s_x + 2} \quad (4.3)$$

$r_x$  and  $s_x$  are the collected number of satisfactory and unsatisfactory transactions by  $x$  and  $Rep_x$  is the reputation score. The model also supports forgetting factor and two operators for combining and discounting opinions. Forgetting factor is to give recent transaction more weight. The purposes of combining and discounting operations are respectively to combine opinions from different sources and discount opinion received by a chain of advisors.

## 4.4 Modeling uncertainty

Uncertainty represents the degree to which a piece of information is close to the true value in the real world. The one that interests us in our work is the uncertainty due to lack of trust on the information source (or inconsistent information) and uncertainty due to imprecise nature of the given information (or incomplete information).

As discussed in the state of the art white paper on managing uncertainty in section 3.2.1, there are various types and sources of uncertainty. There are several uncertainty models. The choice of a model depends on the nature of data that is handled. For the Intelligent community use case, we find that the major problems of uncertainty are related to information inconsistency arising from multiple sources, and imprecision in the shared information (e.g., "Soil is Dry" information shared by a neighbor is imprecise as degree of dryness of the soil is not shared). An uncertainty model that suits such information is *Possibilistic Logic* as it cannot only capture extreme forms of partial knowledge like complete knowledge to total ignorance about an information. We discuss this in the subsection 4.4.1.

The data from Smart City Garbage collection use case on the other hand exhibits different nature of uncertainty. The information in the use case is mostly precise as we already have logs of different drivers and garbageman collecting the garbage from different locations in the city. The uncertainty issue in this use case is to determine what course of action needs to be taken from various reports of the drivers. Hence, we use *Conditional Probability* model to determine the certainty of different actions to be taken. We discuss this in the subsection 4.4.2.

These models of uncertainty provide a mechanism of propagating uncertainty from low level information to a high level one. We further need a mechanism to compare and decide amongst the different high level information that is reasoned. We explain this in subsection 4.8.

### 4.4.1 Possibilistic Logic

Possibility theory is an uncertainty theory that is useful for the handling of incomplete information. The classical uncertainty models such as the probability theory is not able to capture partial ignorance about the information. As will be seen from the definition, the theory makes it possible to model uncertainties in the extreme cases. It differs from probability theory by the use of a pair of dual set-functions called the *possibility* and *necessity* measures instead of only one.

The possibility theory represents uncertainty by two measures: possibility and necessity. A possibility distribution is a mapping  $\pi$  from a set of states of affairs  $S$  to a totally ordered scale such as the unit interval  $[0,1]$ . The function  $\pi$  represents the knowledge of an agent (about the actual state

of affairs) distinguishing what is plausible from what is less plausible, what is the normal course of things from what is not, what is surprising from what is expected. It represents a flexible restriction on what the actual state of affairs is, with the following conventions:

- $\pi(s) = 0$  means that state  $s$  is impossible,
- $\pi(s) = 1$  means that state  $s$  is totally possible, but the degree of certainty with which it is possible is given by the necessity.

If the state space is exhaustive, at least one of its elements should be the actual world, so that at least one state is totally possible and there can be more than one value that may be simultaneously possible; given by possibility values being equal to 1. The theory is driven by the principle of *minimal specificity*. It states that any hypothesis not known to be impossible cannot be ruled out. The possibilistic framework can capture the two extreme forms of partial knowledge:

- **Complete knowledge:** for some state  $s_0$ ,  $\pi(s_0) = 1$  and  $\pi(s) = 0$  for all other states  $s$  (only  $s_0$  is possible)
- **Complete ignorance:**  $\pi(s) = 1, \forall s \in S$ , (all states are totally possible).

A possibility theory assigns each element  $s$  of the universe of discourse  $S$ ,  $\pi(u) \in [0, 1]$ . The uncertainty in a possibilistic clause is represented by two measures Possibility ( $\Pi$ ) and Necessity ( $N$ ). For all  $p, q \in U$ , possibility and necessity measures satisfy the following axioms:

1.  $\Pi(\perp) = N(\perp) = 0, \Pi(\top) = N(\top) = 1$
2.  $\Pi(p \vee q) = \max(\Pi(p), \Pi(q))$
3.  $N(p) = 1 - \Pi(\bar{p})$
4.  $N(p \wedge q) = \min(N(p), N(q))$
5.  $N(p) \leq \Pi(p)$

### Why possibility theory?

We chose possibilistic logic to model uncertainty in the information because of imprecise and vague nature of information that can be provided by the information providers in the IoT. E.g., for a proposition that “it will rain tomorrow” provided to an agent  $A$  by a weather station, the necessity of the event occurring can be assumed to be equal to trust on the weather station. Further more, the possibility theory is based on max-min arithmetic which makes the computation simple. The issue with the theory is how do we get the values of possibility and necessity for an event in our domain of application.

### 4.4.2 Probability Theory

In the smart city garbage collection use case, we have a different requirement with regards to modeling uncertainty in the given information. Here, we already have the prior information regarding the various remarks made by the garbage men and the corresponding decisions taken. With this in view, we think a conditional probability model fits this requirement. A conditional probability measures the probability of occurrence of an event given that some other event (an assumption, assertion) has already occurred. This is governed by *Bayes' Theorem*. It is stated as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \quad (4.4)$$

where,  $A$  and  $B$  represent two events.

- $P(A)$  and  $P(B)$  are the probabilities of  $A$  and  $B$  without regard to each other.
- $P(A|B)$ , a conditional probability, is the probability of  $A$  given that  $B$  is true.
- $P(B|A)$ , is the probability of  $B$  given that  $A$  is true.

Our problem in the use case is : “*Given the remarks about the different bins by the garbage men, what is the best action to be taken?*”. In other words, we want the system to propose us different actions with their corresponding uncertainties based on the drivers’ remarks. Let  $C$  and  $R$  represent a conclusion and a remark. If  $C$  can be derived from  $R$ , then  $P(C, R)$ , that is the probability of having the event  $C$  as conclusion and  $R$  as the remark, can be expressed as

$$P(C, R) = P(R|C) \cdot P(C) = P(C|R) \cdot P(R) \quad (4.5)$$

From this equation, the probability of the conclusion can be computed as:

$$P(C) = \frac{P(C|R) \cdot P(R)}{P(R|C)} \quad (4.6)$$

For a given remark  $R = \text{”WrongSorting”}$  which represents the fact that the bin was not sorted properly, the conclusion  $C = \text{”Replan”}$ , meaning there needs to be another pickup planned in the future,  $P(\text{Replan}|\text{WrongSorting})$  is equal to 1. This is because in the use case, remark *WrongSorting* always leads to the conclusion *Replan*. A table showing some remarks along with the corresponding reasoned information is shown in the table 4.2. It shows that the remarks *Wrong Sorting*, *Bin Not Put Forward* and *Additional Waste* lead to the conclusion *Replan*, while the remark *Defective Container* leads to the conclusion *Replace Bin*.

Table 4.2: A sample remarks and reasoned information table

	Wrong Sorting	Bin Not Put Forward	Additional Waste	Defective Container
Replan	5	2	2	0
Replace Bin	0	0	0	6

The corresponding conditional probabilities for the event *Wrong Sorting* given that the conclusion was *Replan* and *Replace bin* can be computed as below.

$$P(\text{WrongSorting}|\text{Replan}) = \frac{5}{5 + 2 + 2 + 0} = 0.55 \quad (4.7)$$

$$P(\text{WrongSorting}|\text{ReplaceBin}) = \frac{0}{5 + 2 + 2 + 0} = 0 \quad (4.8)$$

## 4.5 Agent beliefs and their relation with trust

The basic function in Internet of Things (IoT) is exchange of information amongst different interconnected things. It involves two agents - the information provider and the information receiver. The providers can advertise their ability of providing a specific type of information via a ‘‘Directory Facilitator’’ type of agent that maintains lists of all information providers and their respective services.

A complete interaction where an information provider agent has provided a piece of information to the information receiver agent is called a transaction. At the end of a transaction, each of the involved parties rate each other about the way the transaction was made. The decision to rate each other depends on numerous factors. One of the important factors is the degree of closeness of the information provided by the provider and the actual true value.

### 4.5.1 Interaction

As illustrated in 4.4, we model each interaction (*Ir*) as a 5-tuple.  $Ir = \{Q, R, \alpha, Res, t\}$

- Q = Querying Agent
- R = Replying Agent
- $\alpha$  = Specific type of information being queried
- *Res* = Response provided by R with respect to  $\alpha$
- t = Time instance when the information *Res* is valid

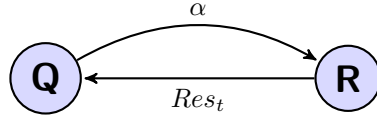


Figure 4.4: A transaction between querying agent  $Q$  and replying agent  $R$

### 4.5.2 Beliefs

Beliefs are propositional statements that an agent considers as being true. An agent develops beliefs on the basis of information received from outside (other agents or the world). At the end of an interaction with an replying agent  $R$ , a querying agent  $Q$  develops a belief on the question  $\alpha$  it queried. A belief is represented as  $Belief(\alpha : Res, t, T_{QR})$ , where  $T_{QR}$  is the quantitative value of trust of agent  $Q$  on agent  $R$  and it is computed by algorithms presented in section 4.3.

An agent develops beliefs in its belief base in one of the following ways.

1. A piece of information arrives from an external source (other agent, sensor etc.). E.g., if a weather information source says that it will rain tomorrow to a home agent, it will develop a belief about weather tomorrow.
2. A new information is inferred out of the available beliefs and the rules in the rule base of the agent. For example, as in Intelligent Community use case, if the agent has a rule that it should send a signal to stop its watering device when it will rain, then the agent will infer a belief that it should send a signal to stop the watering device. We call these beliefs as “inferred” or “derived” beliefs. The process of inference is generally triggered when the agent receives an information from exterior (other agents or sources of information).

We associate a degree of certainty to each of the beliefs. This value is obtained from the trust value of the source from which the belief was derived from. We explain this further in the following section.

## 4.6 Using trust measure as uncertainty

When an information is provided by an information source, the information uncertainty or the degree of truthfulness of the information is unknown often. In such situations, we hypothesize that the certainty or truthfulness of the information is proportional to the trust of the information source at that instance of time. As presented in the section 4.3, the trust models permit us to maintain quantitative measures of trust for various sources at any instance of time. We normalize these trust measures to be within the range

[0,1]. So, the trust measures are relative and an information source with a measure close to 0 is untrustworthy and that close to 1 is highly trustworthy. The advantage of such normalization is that the values can be directly used as measure for uncertainty.

As an example, let us consider *HomeA* and *HomeB* be two homes. Let us assume the trust *HomeA* has on *HomeB* and that *HomeB* provides information that the soil is dry. As shown in the evolution of beliefs below, the applied rule converts the trust on the neighbor to uncertainty in step 7.

This interaction will be encoded in *HomeA*'s ontology as:  
Initial relations in the ontology with regards to trust on *HomeB*.

1. hasIsTrueRelation(HomeB, isTrueRelationHomeB)
2. hasTrustModel(isTrueRelationHomeB, "Eigen")
3. hasTrustValue(isTrueRelationHomeB, 0.5)

After *HomeA* receives "DrySoil" information.

4. says(HomeB, DrySoil)

After a rule *Says(Neighbor, info) ->hasBelievesRelation(Me, info), hasUncertaintyModel(info, "Possibilistic"), hasUncertaintyValue(info, Trust(Neighbor))*

5. hasBelievesRelation(Me, beliefRelationSoil)
6. hasUncertaintyModel(beliefRelationSoil, "Possibilistic")
7. hasUncertaintyValue(beliefRelationSoil, "0.5")

## 4.7 Assumption-based Truth Maintenance System

An Assumption-based Truth Maintenance System is a system for maintaining consistent set of beliefs in the knowledge base of an agent. It is attached to a Rule Engine, which provides inputs to the ATMS in the form of justifications. It maintains a dependency network for different information nodes and their dependencies on a set of assumptions.

For our work, the assumptions are mainly the trust relations of various sources. So, the labels of several nodes, which is a set of environments contain assumptions of trust relations the conjunction of which can derive the nodes. At any instance of time, the agent's ATMS helps it to keep track of how newer beliefs are dependent upon the assumptions of trustworthiness of the various sources. A diagram of how we use ATMS along with the reasoner is illustrated in figure 4.5. For each inferred information, there are set of rules in the rule base of the reasoner that derive it. This set of rules is then fed to the ATMS as justifications. The ATMS algorithms enable us to query for each proposition and the corresponding environments that derive



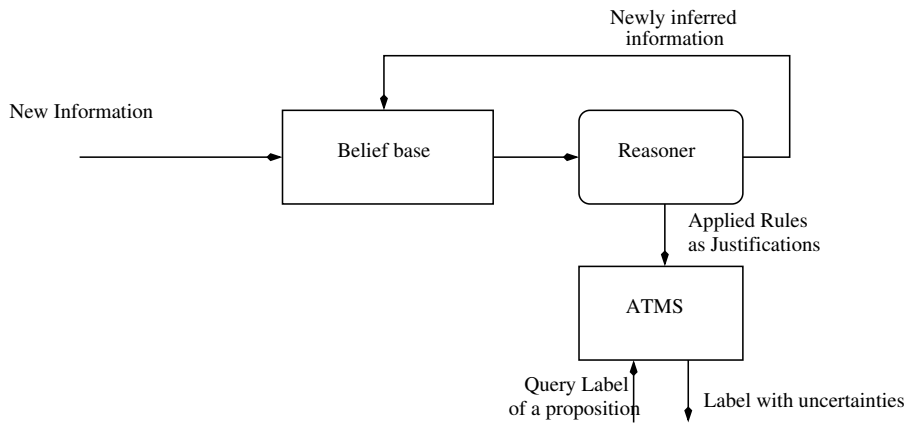


Figure 4.5: Reasoner and ATMS

along with their uncertainty values. The labels of the nodes in the ATMS provide a basis for comparison of different conclusions. This is explained further in the following section 4.8.

## 4.8 Reasoning with uncertainty

The beliefs of an agent can evolve in one of the two ways as explained in the section 4.5.2. At a time instance  $t$ , when the existing beliefs of the agent (*beliefs* –  $t$ ), are actually validated by some mechanism (validation mechanism is specific to domain), the validation outcome can either agree or disagree with a derived belief. The trust update mechanism depends on how the belief was developed (or derived). In general, a belief  $Z$  can be developed as shown in one of the three ways in the figure 4.6. The consideration of more assumptions will just be a duplication of cases (b) or (c) or both (b) and (c). Here,  $A_1$  and  $A_2$  represent two different assumptions about the trustworthiness of the same source (different type of information) or different sources (of the same or different types of information). The analysis of the trust update in each of these cases is explained below and in the table 4.3.

(a) In figure 4.6.a, belief  $Z$  is derived from an assumption belief  $A_1$  as illustrated by the labels of the nodes. The trust update in this case is straightforward. The trust of the involved agent needs to be incremented or decremented according to the validation response agreeing or disagreeing with the belief. This is done by adding a rating to the corresponding transaction between the interacting agents.

(b) The derivation shown in figure 4.6.b, illustrates a scenario where a belief  $Z$  can be derived from a conjunction of two different assumption

beliefs  $A_1$  and  $A_2$ . If the outcome of the validation process of the belief  $Z$  is in accordance with the belief, then this implies both the assumptions are true and since the assumptions are basically related to the trust of the sources their trust ratings can be incremented. Conversely, nothing much can be reasoned in the other case when the decision of validation is not in accordance with the belief since which of the two assumptions is false is unknown.

(c) In the case depicted by figure 4.6.c, the belief  $Z$  is derived from disjunction of assumptions  $A_1$  and  $A_2$ . Like in case (b), reasoning is not possible when the decision of validation is accordance with the belief as it is not possible to reason which of the two assumptions  $A_1$  or  $A_2$  were true.

---

**Algorithm 3:** Algorithm for update of trust

---

$t$  = the current time instance

$beliefs$  = set of all beliefs of an agent

$beliefs-t$  = subset of beliefs that need validation at time  $t$

$t$  = a variable that stores the current value of time passed since the agent came into existence

$transactions$  = a set of transactions related to  $beliefs-t$

▷ Obtain  $beliefs-t$ .

At a time instance  $t$ , obtain a subset of  $beliefs$  that need validation at that time.

**for each**  $b$  **in**  $beliefs-t$  **do**

Validate truthfulness of question  $\alpha$  and response  $Res$

Update the transaction(s) corresponding to  $b$  with a rating of +1 or -1 depending upon whether the validation step was acceptable or not.

**end**

---

The mechanism of update of trust is explained in the algorithm 3. The algorithm begins by computing the current time instance  $t$ . This is followed by obtaining  $beliefs-t$ , a subset of all the  $beliefs$  of the agent that need validation at time  $t$ . For each of the beliefs in  $beliefs-t$ , the agent validates it. The validation process either agrees with the belief, in which case the trust on the source of the belief is incremented and vice versa. However, as explained in the cases (b) and (c) above, the agent may arrive in a situation where it is unable to reason about which of the sources of information was responsible for an inferred belief. In such situation, the agent needs to further explore and validate the individual assumptions to reason which of the sources are true.

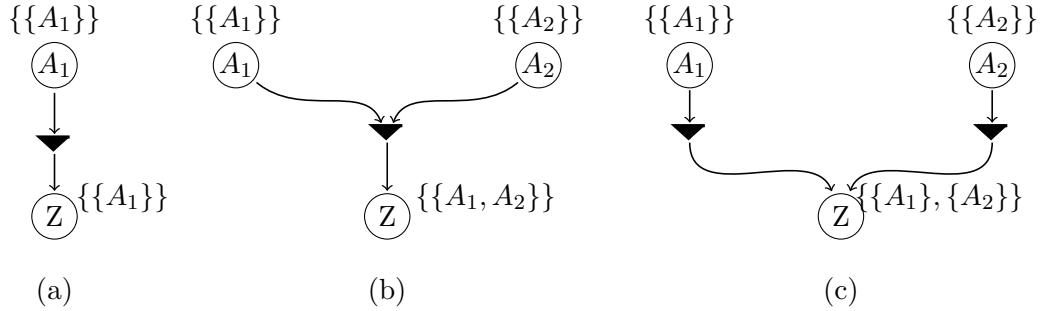


Figure 4.6: Dependency graph snapshots from the ATMS depicting possible cases in reasoning trust relations.  $A_1$ ,  $A_2$  are assumption nodes while  $Z$  is a derived node. The corresponding labels of the nodes are shown alongside.

## 4.9 Discussion

This chapter presented our approach to resolve the issues of modeling and management of trust and uncertainty in distributed systems of agents. We presented two trust models; namely, Eigen Trust and  $\beta$ -Reputation, and two uncertainty models; namely, Possibilistic logic and Probability theory. The initial beliefs are associated with quantitative value of uncertainty depending upon the trust value of the source that furnished the information. The reasoner executes the domain specific rules and the uncertainty model propagates the uncertainty from propositions to the conclusion(s). The ATMS coupled with the reasoner enables to know in how many ways a conclusion may be derived and their corresponding uncertainty values. This allows us to pick the events/actions that are more certain than others. Overall, we presented the architecture from a particular agent's perspective. With multiple agents, this behavior can be replicated as we show in the following chapter.

Table 4.3: Analysis of derivation of conclusion

<b>Label of belief Z</b>	<b>Validation response</b>	<b>Remarks</b>
$\{\{A_1, A_2\}\}$	Agree	Trust of both the sources of information needs to be incremented
$\{\{A_1, A_2\}\}$	Disagree	Either $A_1$ or $A_2$ or both $A_1$ and $A_2$ are false. Hence, trust for which source needs to be updated is not clear.
$\{\{A_1\}, \{A_2\}\}$	Agree	Either $A_1$ or $A_2$ or both $A_1$ and $A_2$ are true. Hence, trust for which source needs to be updated is not clear.
$\{\{A_1\}, \{A_2\}\}$	Disagree	Trust of both sources corresponding to $A_1$ and $A_2$ need to be decremented.



## Chapter 5

# Implementation

This chapter discusses the implementation details of our approach. A generic implementation for the use cases of our work consists three main aspects: (i) Distribution aspect, (ii) The agent and the (iii) The data representation. We discuss them in the sections 5.1, 5.3 and 5.2 respectively.

To further explain the sections and subsections of the chapter, we make use of the following example from the Intelligent Community use case. As depicted in figure 5.1, we consider two homes *A* and *B* in a neighborhood. Homes *A* and *B* are *smart* meaning that they are equipped with devices connected to the Internet and thereby benefiting from the services of different service providers. A typical example of such a service is obtaining weather information from weather forecast companies. Depending upon the information the devices can alert the users, or take a decision to activate an actuator. We further assume that home *A* has no further sensor attached as a source of information and thereby has to rely solely on the weather forecast (particularly the chance of rain) to control the watering systems. Home *B*, on the other hand, has sensors such as a soil moisture detector and a humidity sensor deployed for it's use in addition to the weather information from the weather forecast company. Furthermore, the owner of home *B* is open to sharing soil moisture and humidity readings with his/her neighbors.

### 5.1 Distribution aspect

As we explained in section 4.1, we envision a distributed environment as a MAS and the entities of the system as MAS agents. There is an important number of implementations in agent programming framework, based on Belief Desire Intention (BDI), like BDI4JADE, JADEX, 2APL. However, we disregarded this domain as we found that BDI architecture does not fit domains that need learning and adapting their behavior [GPP<sup>+</sup>98]. Though our domains are not completely related to learning, there is a need for updat-

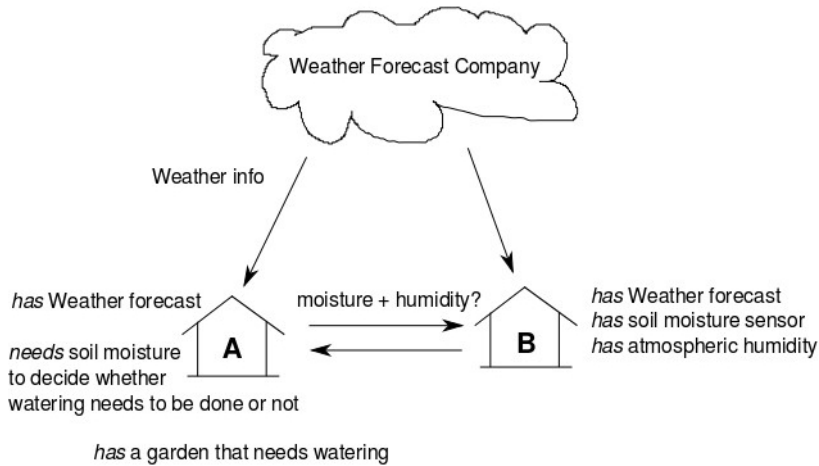


Figure 5.1: Example from the Intelligent community use case.

ing the trust of the various sources and adapting to their new values. Hence, we use JADE, which is a robust and a highly customizable implementation framework for a MAS. It is a standard framework for the implementation of MAS. It is a stable, open source and widely known. It supports a number of FIPA communication protocols and is highly modular and customizable for the needs of a research. Once we define our agents, their behaviors and the components they contain, we can easily instantiate them to form a MAS.

JADE provides an environment called “*container*” within which programmable agents can be created. Agents forming a MAS can be distributed amongst multiple containers for scalability. An illustration of the use of JADE for the Intelligent Community use case is shown in the figure 5.2. Here, the agents *HomeA*, *HomeB* and *HomeC* exist in *CONTAINER-1*, while *HomeD* and *WeatherInfoProvider* form the part of *CONTAINER-2*. The interactions within agents of the same or different containers is facilitated by JADE. The arrows indicate the interactions between different homes.

## 5.2 Data representation

We use Linked Open Data to represent the data and the knowledge base of the agents. The data model used for Linked Data is called Resource Description Format (RDF). It is a structured way of representing information with classes and their relations. An ensemble of such a data set is called an Ontology. We developed an ontology (TUM-ontology) that defines the concepts and the relationships used for management of trust and uncertainty (TUM). Each of the agents that need the trust and management module, share this

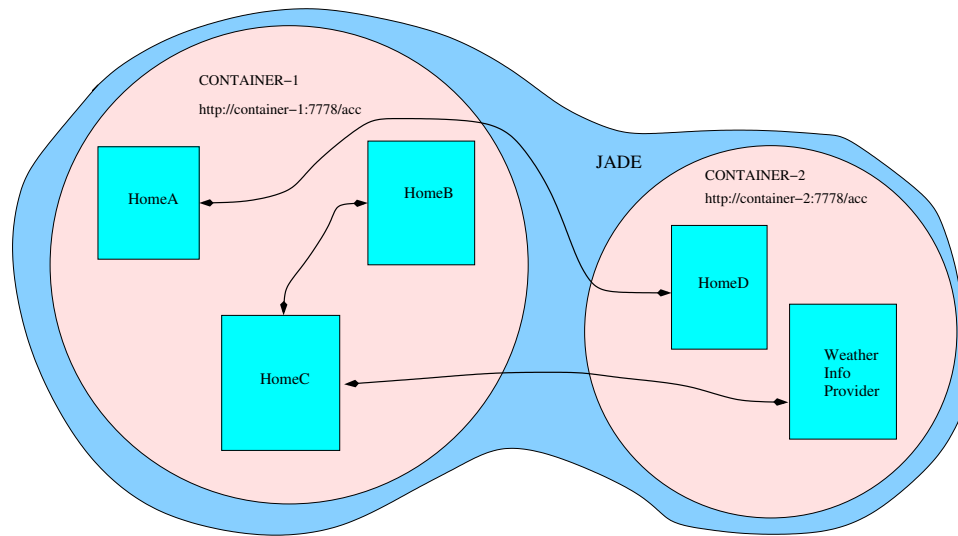


Figure 5.2: JADE facilitating communication amongst agents in two containers.

ontology. However, their contents differ based upon the application domain and the evolution of their knowledge bases. The ontology also includes the domain specific rules that the agents use for derivation of new inferences. The rules are modeled in Semantic Web Rules Language (SWRL), as it is compatible with RDF.

The complete ontology is depicted in the figure 5.3. The major classes of the ontology are explained below.

1. **Agent:** This class represents the entities of the distributed system under consideration. **InformationSource** is a subclass of this class, that represents those agents that are providers of the information. For the example in 5, *HomeB* can be an **InformationSource** for *HomeA*.
2. **Information:** For an agent, the messages received from other agent(s) are information. We store them as instances of the class **Information**. *DrySoil* and *WetSoil* that represent the soil conditions can be examples of this class.
3. **InformationTopic:** We need to categorize the information as a source of information can have different levels of trust for different categories of information. We represent this using the class **InformationTopic**. E.g., *WeatherInfo* and *SoilInfo* are be two different information topics used in the example.
4. **isTrue\_Relation:** This is a reified class to represent that an information source is trustworthy with respect to a particular **InformationTopic**. It is related to two other classes **TrustModel** and **TrustMeasure**.



5. **TrustModel**: This class represents the model of trust used for representing the trust of the information sources. We used *Eigen* trust in our work, hence it is the only subclass of this class. Any other model for trust will be a subclass of this class.
6. **TrustMeasure**: This class is used for storing the numeric value of trust for the sources based upon the model of trust used. **EigenTrustMeasure** is a subclass of this class.
7. **Uncertainty**: The uncertainty in the information is represented by this class. The two models of uncertainty that we studied in our work form the two subclasses of this class **ProbabilityUncertainty** and **PossibilityUncertainty**.
8. **UncertaintyModel**: Uncertainty in the information is modeled using a model, that is represented by this class. It has two subclasses **ProbabilityModel** and **PossibilityModel**.
9. **ConditionalProbability**: This class is used to model the conditional probability discussed in subsection 4.4.2.

The relationships between the different classes marked in blue in the figure are explained below.

1. **believes**: This relation encodes the fact that an agent believes an information.
2. **says**: We use this relation to represent that an exterior agent provided an information to the agent under consideration. The agent under consideration is implicit.
3. **isAboutTopic**: An information may be related to various categories or topics. As we compute trust of an agent with respect to the topic of the information, we need this relation. It relates **Information** to **InformationTopic**.
4. **has\_isTrue\_Relation**: In order to represent an information like: “ $X$  is trustworthy with respect to  $T$  type of information and the value of  $V$  and a trust model  $M$ ” in terms of triples, we need an intermediary class and intermediary relations. This is called *Reification*. We created the class **N-ary\_Relation** to represent a reified relation. Thus, the relation **has\_isTrue\_Relation** is the equivalent of one of the intermediary relations above. It relates an **InformationSource** to an **isTrue\_Relation**.
5. **isTrueAbout**: An **InformationSource** may be trustworthy with regards to different types (topics) of information. The **isTrueAbout**

relation indicates, to what type of information (`InformationTopic`) an `InformationSource` is trustworthy about.

6. `hasTrustMeasure`: This relation relates the reified `isTrueRelation` to the `TrustMeasure` class, which holds a numeric measure for the trust.
7. `hasTrustModel`: This relation indicates what model of trust the `isTrueRelation` is possessing.
8. `hasUncertainty`: We quantify the uncertainty in a given information by `hasCertainty` relation.
9. `hasUncertaintyModel`: We modeled the uncertainty in the information using two models `ProbabilityModel` and `PossibilityModel`. This relation specifies what uncertainty model does an `Uncertainty` have.
10. `hasConditionalProbability`: This relation relates further an uncertainty of `ProbabilityModel` to `ConditionalProbability`. By definition, the `ConditionalProbability` class has two relations associated with an information. They are modeled using the `hasInformation` and `hasGivenInformation` relations.

Example: The following expression shows how an agent encodes the information that *B* informs *A* about the weather and the corresponding uncertainty of the information. Here `B`, `Rain`, `Weather`, `u1`, `Possibility` are the individuals of `InformationSource`, `Information`, `InformationTopic`, `Uncertainty` and `PossibilityModel` classes of the ontology respectively. `hasUncertaintyValue` is a data property of the `Uncertainty` which stores the quantitative value of the uncertainty of the information.

```
says(B, Rain),
isAboutTopic(Rain, Weather),
hasUncertainty(Rain, u1),
hasUncertaintyModel(u1, Possibility),
hasUncertaintyValue(u1, 0.65f)
```

### 5.3 The agent

The agent is an entity of the MAS. We model an agent consisting of different behaviors and components as illustrated in the figure 4.1. JADE requires the agent to be a subclass of the `jade.core.Agent` class from the JADE api. For an agent to accomplish a certain task, they need to be defined. In JADE,

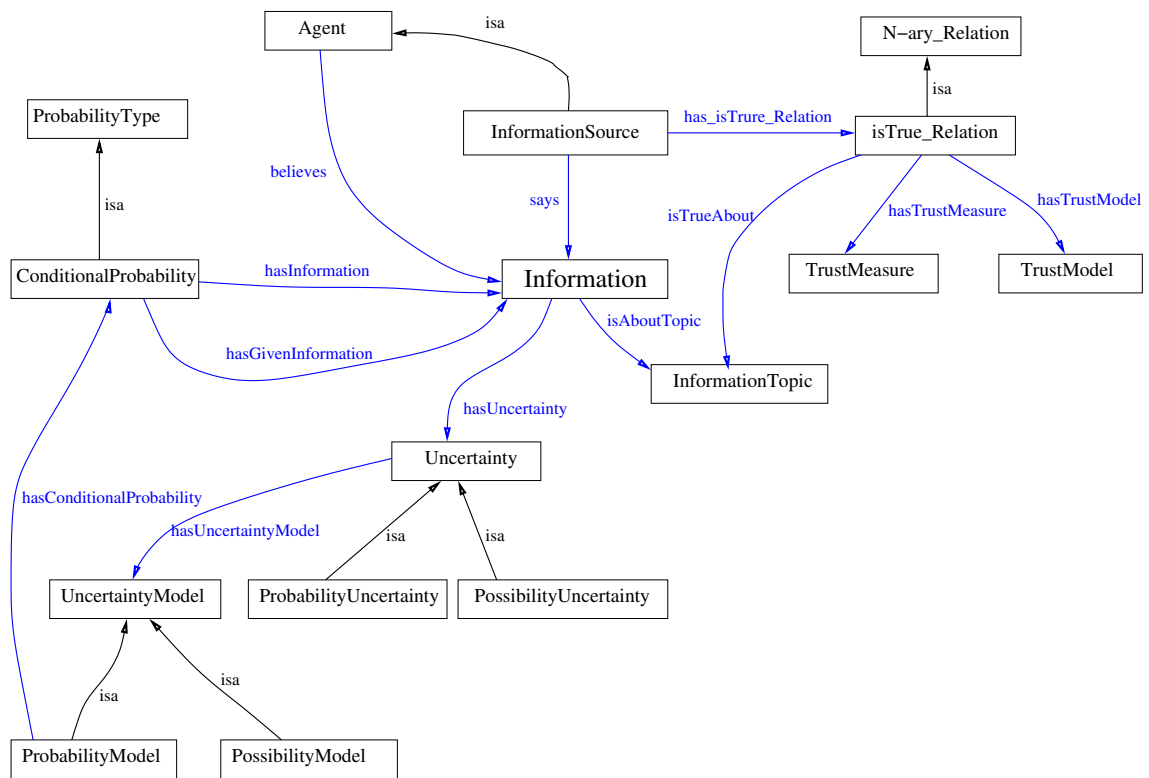


Figure 5.3: TUM ontology.

a task is called a *behavior* and is defined by extending the subclasses of `jade.core.behaviours`. Several different types of behaviors are supported by JADE like `CyclicBehaviour`, `OneShotBehaviour`, `ParallelBehaviour` etc. For our purposes, `OneShotBehaviour`, which models the behavior to be executed just once, is sufficient. Hence, the behaviors of the agent we modeled, use `OneShotBehaviour`. Apart from them, the agent includes the reasoner, the rule base, the ATMS, and the transactions and the trust table. We discuss in detail, the behaviors of the agent in subsection 5.3.1, the rule base in subsection 5.3.3, the reasoner in subsection 5.3.2 and the implementation of the ATMS in subsection 5.3.4 respectively.

### 5.3.1 The agent behaviors

An agent may have different behaviors depending upon the application domain. For the domains that concern our work, we model the agent to have three main behaviors: Message handling behavior, Validation behavior and the belief update behavior. An illustration of how these behaviors fit into the architecture of the agent is shown in the figure 4.1. Below, we explain each of these behaviors with examples.

1. Message handling behavior: The message handling behavior deals with the incoming messages, transforming them into *Beliefs*. The behavior either uses the belief base to construct and respond to the request in the messages or uses the response to form a belief. The information exchange is based on FIPA query interaction protocol. The agents seeking information do a query with a `query-ref` message, seeking a specific information. The responding agent either accepts and sends the `inform-result` or rejects the request. As we represent the data in terms of RDF, we use SPARQL CONSTRUCT query string as content in the query message. The `inform-result` message contains an RDF.
2. Validation behavior: The validation behavior is related to management of trust of the sources of information. At regular intervals of time, an agent validates whether the beliefs in its belief base are true. Depending upon the outcome of validation, the agent either increments (if the belief was true) or decrements (if the belief was false) the trust of the information source that furnished the information.
3. Belief update behavior: This behavior allows the agent to infer new beliefs from the current set of beliefs. The inferred information and their respective uncertainty is obtained from the reasoner and the ATMS. At the end of each inference, the reasoner inserts the explanations for the inferred information and the uncertainties into the ATMS, which then detect the inconsistencies in the derivation. The outcome of the ATMS is then used to update the beliefs of the agent.

### 5.3.2 The reasoner

The reasoner infers new information from a set of rules and given information. There are several implementations of reasoners available. But, when we began our work, the criteria for selection of a reasoner for us was: (i) it should be open source, preferably Java (for easy integration), (ii) it should be semantic, i.e., able to handle data in RDF format and (iii) it should be able to understand rules from a standard language such as Semantic Web Rules Language (SWRL). So, with these criteria, we filtered out between Pellet and FaCT++. We chose Pellet for its good documentation and availability as plugin in standard RDF managing tool, *Protégé*. Also, the Pellet reasoner functions well rules in SWRL. Unfortunately, the creators of Pellet have recently declared the latest version (Pellet 3.0) as closed source.

We use the Pellet reasoner for two main functions. First, inferring new information from the existing set of rules and given facts/axioms. Second, obtaining the explanations for the inferred information. The belief validation behavior inputs these explanations to ATMS as justifications for the inferred information. All the relations in the ontology are *premises* for the ATMS except for the `isTrueAbout` relation. Based on the uncertainty model, the uncertainty propagation of the information is included within the rules. Hence, the inferred information from the reasoner is associated with a corresponding value of uncertainty.

#### Example

If the reasoner applies the rule presented in section 5.3.3, `WateringDeviceStatus`, which is an individual of the class `Information` in the ontology, is inferred. The Pellet reasoner API permits us to obtain the explanation for the inferred information. For `WateringDeviceStatus`, the following explanations are provided by the reasoner. As shown in the list of explanations given below, the necessity of the inferred information `WateringDeviceStatus` is `u1`, which has an uncertainty value of 0.7.

```
Individual: <http://www.semanticweb.org/lpkc4220/ontologies/2015/3/
  untitled-ontology-35#WateringDeviceStatus>
-- DataPropertyAssertion : hasStatus(WateringDeviceStatus, false)
Explanations are:
  - believes(A, RainyWeatherInfo)
  - hasWateringDevice(A, DeviceA)
  - hasUncertainty(RainyWeatherInfo, u1)
  - hasUncertaintyValue(u1, 0.70)
- believes(A, RainyWeatherInfo) ^ hasWateringDevice(A, ?
  wateringDeviceOfA) ^ hasUncertainty(RainyWeatherInfo, ?unc) ^
  hasUncertaintyValue(?unc, ?alpha) ->
  hasDeviceStatus(?wateringDeviceOfA, WateringDeviceStatus) ^
  hasUncertainty(WateringDeviceStatus, ?uncDerived) ^
  hasUncertaintyValue(?uncDerived, ?alpha) ^
```

```
hasStatus(WateringDeviceStatus, false)
```

### 5.3.3 The rule base

The rule base stores the rules of an agent. Semantic Web Rules Language (SWRL) is a standard put forward by a W3C committee for expressing rules and logic for the semantic web. SWRL is XML-based and is not human readable. We need a software that can parse, code and encode a human readable rule into XML. We created SWRL rules within the ontology using Protégé which provides a prolog-like human-readable input interface and converts it to SWRL compatible XML. The XML encoded rules are stored within the ontology itself. The Pellet reasoner is able to pick rules from the ontology. An example of one such rule in the rule base of *A* in the intelligent community example is given below. The rule represents the fact that if the agent *A* believes an "RainyWeather" information with an uncertainty, it can derive the fact that it needs to switch off its watering device with the same uncertainty. Here, the terms that precede with a question mark represent a variable, e.g.: `?wateringDeviceOfA` is a variable that may be replaced with a real instance from the ontology by the reasoner. Also, we notice that the rule has a new relation `hasDeviceStatus`. It relates a watering device type of object to the status it may have. This is specific to the domain of Intelligent community application and hence it does not appear in the generic ontology explained earlier.

```
believes(A, RainyWeatherInfo) ∧
hasWateringDevice(A, ?wateringDeviceOfA) ∧
hasUncertainty(RainyWeatherInfo, ?unc) ∧
hasUncertaintyValue(?unc, ?alpha) →
hasDeviceStatus(?wateringDeviceOfA, WateringDeviceStatus) ∧
hasUncertainty(WateringDeviceStatus, ?uncDerived) ∧
hasUncertaintyValue(?uncDerived, ?alpha) ∧
hasStatus(WateringDeviceStatus, false)
```

### 5.3.4 The ATMS

As we explained in section 3.4, the ATMS serves to maintain a consistent set of beliefs for the agent and also obtain explanations for why a data node is true or false. The ATMS we used in our implementation was developed from scratch in Java. Though numerous implementations of DeKleer's ATMS exist in LISP, to the best of our knowledge, none was available in Java. We developed this ATMS in Java in order to remain consistent with a single programming language for the entire project. The implementation is based on DeKleer's paper [DK86]. The key function of ATMS is to maintain a consistent set of labels for its different nodes. A true information is represented by a node that is called a *premise*. It is represented by a label with

empty set  $\{\{\}\}$  meaning it can be derived by itself and does not depend on any *assumption(s)*. A non-empty label represents the environments that can derive the node. E.g., a node with the label  $\{\{\text{Rain, DrySoil}\}, \{\text{NoInfo}\}\}$  signifies that it is derivable under the condition  $\text{Rain} \wedge \text{DrySoil}$  or  $\text{NoInfo}$ . When a new data is inserted into the ATMS, the **weave** and **propagate** algorithms from [DK86], ensure that the labels of all the nodes are updated into order to maintain the logical properties like consistency, soundness, minimality and completeness. The various important terminologies used in the ATMS are described in section 3.4.1.

We added the ATMS implementation to each of the agents in of the distributed system for maintenance of each of the agents. A class diagram of the implementation of the ATMS is shown in figure 5.4. As shown in the figure, the ATMS API consists of six main Java classes that are described below. The getters and setters have obvious meanings of getting and setting an item and hence have been avoided.

1. **ATMS**: It contains all the functions pertaining to declaring the **Justification(s)**, **Contradiction(s)** in the system. The class comprises of lists of **TMSNode(s)**, **Justification(s)**, **Assumption(s)**, **Contradiction(s)**. All of them are indexed, so that each of their new item can be referred by a **counter** value. Apart from these lists, it also contains two hashtables **envTable** and **nogoodTable**. They store environments and nogood environments respectively based on their numbers, i.e., an environment with two assumptions is located in list of the hashtable with a key "2".
2. **Justification**: A **Justification** defines how a **TMSNode** can be derived from other **TMSNode(s)**. It consists of three parts: (i) a list of antecedent **TMSNode(s)**, an **Informant** and a consequent **TMSNode**.
3. **Environment**: In ATMS terms, an environment is a conjunction of assumptions. The **Environment** class implements this notion and contains a list of assumptions. A *NoGOOD* or an inconsistent environment is a special kind of environment that derives falsity.
4. **Contradiction**: A **Contradiction** represents an inconsistent **TMSNode**. Upon discovery of a contradiction, the ATMS marks all environments containing this contradiction and the algorithms (**weave** and **propagate**) are rerun to obtain minimal consistent environments and reconstruct the labels of various **TMSNode**.
5. **TMSNode**: A **TMSNode** is the basic element of the ATMS. It encapsulates the information of a domain. The information is stored as a **String** variable. An **Assumption** is a special kind of **TMSNode**.
6. **Informant**: This is a description about how a **TMSNode** is derived from other **TMSNode(s)**. We use a **String** variable to represent it.

## 5.4 A working example

In this section, we explain the functioning of the agents with a working example. We take the same example described earlier in the chapter and explore how the belief base and the ATMS are updated from *A*'s perspective. We assume that there are two interactions that occur in sequence: (i) *A* inquires the weather station (say *M1*) about weather condition and receives a response from it. (ii) *A* asks *B* about the soil and the weather information and receives information from it. We show the contents of the belief base and the ATMS at the initial state and after each with the weather provider *M1* and the neighboring home *B*. They are shown in the tables 5.1 and 5.2 respectively. The rules applied during each step, are mentioned in the lower row of the corresponding table. The contents shown in green indicate that they were created as a result of the information originating from an external source, and those shown in blue specify that they were inferred by the reasoner out of the available information and the rules in the rule base.

In the initial state, the ATMS is empty as no reasoning process has occurred. As shown in the table, the belief base consists of a number of triples. We assume that the home *A* has some initial values of trust for homes *B* and the weather provider *M1*; they are:  $M1\text{-weather}=0.8$ ,  $B\text{-soil}=0.85$ ,  $B\text{-weather}=0.55$ . These values are in fact, obtained by applying the trust model to the trust table. The *isTrueAbout* relation is an ATMS assumption, as we are uncertain of its truth. The numbers alongside each of the assumptions shown in red, are the trust values of the respective sources of the assumptions. Home *A* has a programmatically controllable watering device denoted by `WateringDeviceOfA`. All these statements are represented by the triples: *isTrueAbout*(*M1*, *Weather*), *isTrueAbout*(*B*, *Weather*), *isTrueAbout*(*B*, *SoilInfo*) and *hasWateringDevice*(*A*, `WateringDeviceOfA`) respectively. We use Eigen Trust model and Possibility theory for management of the trust and the uncertainty in this example. We obtain the values from the trust table applying the Eigen Trust algorithm, and as explained in the section 4.6, we assume this to be equal to the necessity of the assumption for the possibilistic model of uncertainty propagation.

After communication with the weather provider *M1*, we assume that the home *A* receives the information "It will rain". It is encoded into a triple as *says*(*M1*, *RainyWeather*). The *MessageHandlingBehavior* then fires the reasoner, which then checks if any of the rules in the rule base are applicable. In this case, the rules **R2** and **R3** are applied. As a result, home *A* infers that it will rain and it needs to stop the watering device. This is represented by the triples *believes*(*A*, *RainyWeather*) and *hasStatus*(`WateringDeviceOfA`, *false*). The inferred triples and their explanations (how the triples were inferred) are inserted into the ATMS as TMSNodes and their Justifications respectively. The ATMS contains "M1 isTrueAbout Weather" in ellipse indicating that it is an *assumption*; the others in rectangle specify that they





Figure 5.4: ATMS class diagram.

are *premises*. The relation *isTrueAbout* is an assumption as we are not certain about the truthfulness of the source. The necessity associated with the inferred triples is obtained by possibilistic resolution as described in the axiom 4 of the section 4.4.1.

For an ideal “Intelligent Community” type system, we assume that there is a “service registrar” type of agent in the system. We make two assumptions about this agent: (i) It discovers new homes and the types of services they can offer (ii) It can rank and recommend the querying agent, the neighbors based on their global trust values for their services and their separation (distance). We assume that the recommended agent for home *A*, in this example, is home *B*. So, in the next step, the home *A* inquires home *B* for soil and weather conditions. It receives *DrySoil* and *NoRain* information from *B* and hence it infers *believes(A, DrySoil)*, *believes(A, NoRainyWeather)*, *hasStatus(WateringDeviceOfA, true)* with necessities 0.85, 0.55 and 0.55 respectively using the possibilistic resolution. The algorithm of the ATMS yields two environments that are NOGOOD or inconsistent. They are: {*isTrueAbout(B, Weather)*, *isTrueAbout(M1, Weather)*} and {*isTrueAbout(B, Weather)*, *isTrueAbout(B, SoilInfo)*, *isTrueAbout(M1, Weather)*}. This suggests that the agent has a conflict in its knowledge base.

The conflicting decisions proposed by the system are to switch on and to switch off the watering device which are represented by *hasStatus(WateringDeviceOfA, true)* and *hasStatus(WateringDeviceOfA, false)*. In the application, we propose the conflicting actions: Watering Device ON and Watering device OFF, with their corresponding certainty values to the user and ask for a decision to be taken. In domains, where we need to resolve the conflict and provide a single proposition, we program the agent such that it removes the triple with the least necessity, which signifies removal of the least certain of all assumptions we had initially. For the example above, it would be the removal of the triple *isTrueAbout(B, Weather)* as it has the least necessity associated with it, 0.55. The ATMS is reinitialized with the other assumptions and premises, and a reduced set of decisions and higher necessity values are obtained. This is continued till we have a single and the most certain decision to propose.

The update of trust depends upon the validation step, i.e., the response of the user for the proposed set of decisions. For example, above the agent proposed the user two decisions: Watering device ON and Watering device OFF with necessities 0.55 and 0.8 respectively. If the user chooses to go with the more certain decision and switches off the device, the trust of the sources of the assumptions of the label of *hasStatus(WateringDeviceA, false)* is incremented. Here, the label is *M1 isTrueAbout Weather* and it has a single source *M1*. Hence, the result is increment of the trust of *M1* for weather. Similarly, the sources of the label of *hasStatus(WateringDeviceA,*

true) is decremented as it is against the decision taken by the user. The label of `hasStatus(WateringDeviceA, true)` is `B isTrueAbout SoilInfo`, `B isTrueAbout Weather`. Here, the source of the assumptions is home  $B$ , and there are two assumptions. As explained in section 4.8, we cannot distinguish which of the two assumptions is false. Hence, we ignore decrementing the trust levels of bad sources in this case.

Table 5.1: The initial and after communication with *M1* states of the belief base and the ATMS of home *A*

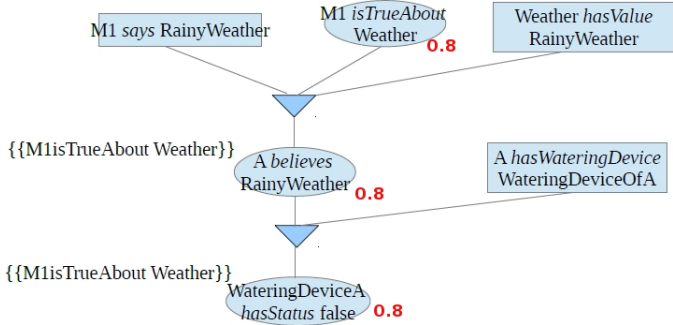
State	HomeA
Initial	<b>Belief base</b> isTrueAbout(M1, Weather) <b>0.8</b> isTrueAbout(B, Weather) <b>0.55</b> isTrueAbout(B, SoilInfo) <b>0.85</b> hasWateringDevice(A, WateringDeviceOfA)
	<b>ATMS:</b> Empty
After communication with the weather information provider <i>M1</i> for weather forecast input	<b>Belief base</b> isTrueAbout(M1, Weather) <b>0.8</b> isTrueAbout(B, Weather) <b>0.55</b> isTrueAbout(B, SoilInfo) <b>0.85</b> hasWateringDevice(A, WateringDeviceOfA) says(M1, RainyWeather) believes(A, RainyWeather) <b>0.8</b> hasStatus(WateringDeviceOfA, false) <b>0.8</b>
	<b>ATMS</b> 
<b>Rules applied</b> <b>R2:</b> hasValue(?informationtype, ?information), isTrueAbout(?source, ?informationtype), says(?source, ?information) $\Rightarrow$ believes(A, ?information) <b>R3:</b> believes(A, RainyWeather), hasWateringDevice(A, ?wateringDeviceOfA) $\Rightarrow$ hasStatus(?wateringDeviceOfA, false)	

Table 5.2: The state of the belief base and the ATMS of home *A* after communication with the neighbor *B* for weather and soil information.

HomeA	
<b>Belief base</b>	
isTrueAbout(M1, Weather)	0.8
isTrueAbout(B, Weather)	0.55
isTrueAbout(B, SoilInfo)	0.85
hasWateringDevice(A, WateringDeviceOfA)	0.8
says(M1, RainyWeather)	
believes(A, RainyWeather)	0.8
hasStatus(WateringDeviceOfA, false)	
says(B, DrySoil)	
says(B, NoRain)	
believes(A, DrySoil)	0.85
believes(A, NoRainyWeather)	0.55
hasStatus(WateringDeviceOfA, true)	0.55
<b>ATMS</b>	
<b>Rules applied</b>	
<b>R1:</b> believes(A, DrySoil), believes(A, NoRainyWeather), hasWateringDevice(A, ?wateringDeviceOfA) $\Rightarrow$ hasStatus(?wateringDeviceOfA, true)	
<b>R2:</b> hasValue(?informationtype, ?information), isTrueAbout(?source, ?informationtype), says(?source, ?information) $\Rightarrow$ believes(A, ?information)	
<b>R3:</b> believes(A, RainyWeather), hasWateringDevice(A, ?wateringDeviceOfA) $\Rightarrow$ hasStatus(?wateringDeviceOfA, false)	
<b>Contradiction rules</b>	
<b>R4:</b> hasStatus(?wateringDeviceOfA, false), hasStatus(?wateringDeviceOfA, true) $\Rightarrow$ false	
<b>R5:</b> believes(A, RainyWeather), believes(A, NoRainyWeather) $\Rightarrow$ false	

# Chapter 6

## Applications

In this chapter, we illustrate the proposed theory in three application domains. The first was is called the *Intelligent Community* that we describe in detail in section 6.1. The second called the *Smart City Garbage Collection* is presented in the section 6.2. The third is an extension of the work to a scientific project called the FIWARE. It is presented in the section 6.3. For each of the application domains we perform a set of experiments. We explain them in detail in the following sections. The third application domain is an extension to an existing project called FIWARE as a generic trust and uncertainty management (TUM) module. We explain this in detail in the section 6.3

### 6.1 Intelligent Community

In this section, we present our experiments with regards to the Intelligent Community use case, illustrated in 2.4.1. As described in the use case, we have a number of a number of homes in the community that need to modeled as agents. These homes are have different sensors for the information on the weather conditions and the soil humidity. The use case presents two types of information sources for a home: (i) it's neighboring homes, (ii) the weather information provider. It can query other homes in the neighborhood for soil and weather information and a weather information provider for weather information to predict the chances of rain. Based on inputs from sources a home needs to decide whether to switch on or switch off its garden watering system with a degree of certainty. The status of garden watering device is ON or OFF depending upon the *Rain* and *Soil* conditions as illustrated in the truth table below. These rules form the rule base of the various homes.

$\wedge$	Rain	No Rain
Dry Soil	OFF	ON
Wet Soil	OFF	OFF

We obtain data for the experimentation of the use case from simulation. We describe this in detail in the subsection 6.1.1.

### 6.1.1 Simulation setup

To experiment with our system explained in 2.4, we simulate the use case by varying different parameters. Basically, the simulation represents the generation of transactions amongst the different agents. In order to do an extensive experimentation we define the following parameters for our simulations.

1. The total number of agents in the system,  $m$ .
2. The set of possible query types amongst the different agents,  $\alpha$ . E.g.,  $\alpha \in \{Soil, Weather\}$
3. The agents, their corresponding services and the level of trust for the services. E.g.,  $\{HomeB : Soil, 0.85\}$ ,  $\{HomeB : Weather, 0.55\}$  and  $\{M1 : Weather, 0.8\}$ . The value 0.85 of trust for *HomeA* for Weather service indicates that out of 100 transactions, it would provide 85 correct values. This is how the script generates different transactions.
4. The set of possible responses by the replying agent,  $Res$  for a service. E.g.,  $Res \in \{DrySoil, WetSoil\}$  for  $\alpha = Soil$ ; and  $Res \in \{Rain, NoRain\}$  for  $\alpha = Weather$ .
5. The order in which the transactions occur. E.g., if *HomeA* interacts with *HomeB* and *M1*, the interaction order may be *HomeA*–*HomeB* and *HomeA* – *M1* or the contrary. This parameter defines whether the order in which the interactions occur should be fixed or random. As we explained in 5.4, we assume that an agent knows which other agent to query and the sequences.
6. The total number of transactions amongst the agents.
7. The trust model for update of the trust of the sources and the uncertainty model for the propagation of uncertainty in the incoming information to the uncertainty in the higher level derivations. For the Intelligent Community work, we mostly worked with the Eigen Trust and Possibility theory.

### 6.1.2 Tests

We need to experiment with the use case in the following aspects that were our success criteria at the beginning of the thesis.

Table 6.1: Experimentation trust update

Source	Information type	Trust	Generation of data
$B$	Soil	0.85	85 % of data is true and 15 % of value false
	Weather	0.55	55 % of data is true and 45 % of value false
$C$	Soil	0.60	60 % of data is true and 40 % of value false
	Weather	0.70	70 % of data is true and 30 % of value false

1. The data must comply with our hypothesis that the uncertainty in the information is proportional to the trust of the source that furnished it.
2. The implementation should be *simple* and *generic*.
3. It must be *scalable* in terms of increase in the number of entities in the system.
4. The trust and uncertainty computation algorithms must be *decentralized* from an entity to the other.

In order to comply with our hypothesis, we set the trust values of different sources of information and generate a number of transactions amongst the homes by simulation. We change the trust values (either increase or decrease) of the sources and generate another set of transactions. The system infers decisions based on the simulated transactions and domain rules, and the corresponding uncertainties. Intuitively, the case where the trust values of the sources are higher, the inferred decision must be more certain than the other case where the trust values are lower.

For the verification of *simplicity* and *genericity* of our approach, we compare it with other existing and other solutions for such a domain.

For verifying the *scalability* of the approach, we increment the number of agents in the experiment and also fix the various interactions and their sequence. Then, we verify if the approach is robust enough to handle these changes.

To verify, the fourth point that is listed above and deals with *distribution* of algorithms, we verify if distributed trust management algorithms such as the Eigen trust can be fitted within the different entities of the distributed system or not.



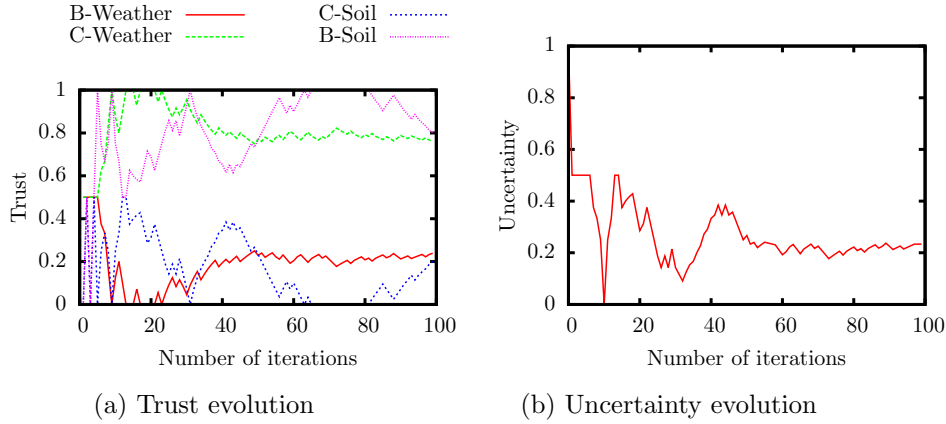


Figure 6.1: Evolution of trust on two sources  $B$  and  $C$ , as computed by  $A$ , and uncertainty of the decision taken for the corresponding iterations

### 6.1.3 Experimentation

In this subsection, we present the different experiments that we carried out to verify the criteria mentioned above, and discuss their results. The first experiment that we present below is pertaining to the validation of our hypothesis.

#### Verifying hypothesis

For this experiment, we consider an Intelligent Community with three homes  $A$ ,  $B$  and  $C$ . As all homes are the same, validating our hypothesis from an agent is sufficient. So, we do this from the point of view of home  $A$ . We consider that  $A$  has a garden and is fitted with a watering device. It needs information about soil and weather conditions and hence, queries  $B$  and  $C$  for soil and weather types of information. We assume that  $B$  and  $C$  generate true values of soil and weather information with 85% and 55% and, 60% and 70% respectively, as illustrated in the table 6.1.

The rules of deriving the conclusion to whether to switch on or switch off the watering for home  $A$  are shown below. Here, **Watering** and **NoWatering** represent the conditions to switch off and switch on of the device respectively. **Rain**, **No Rain** represent the two possible cases of the weather and **DrySoil**, **WetSoil** represent the possible cases for soil conditions.

Rain  $\rightarrow$  NoWatering  
 No rain  $\wedge$  DrySoil  $\rightarrow$  Watering  
 WetSoil  $\rightarrow$  NoWatering

We fix the sequence of interactions as: B-Weather, B-Soil, C-Weather,

Table 6.2: Statistics after 100 iterations

	% of total decisions	% of decisions suggested without conflict	% of decisions suggested with conflict
<b>Watering</b>	61.00% (61)	26.00% (16)	73.00% (45)
<b>NoWatering</b>	39.00% (39)	46.00%, (18)	53.00% (21)

C-Soil; where B-Weather represents the interaction between *A* and *B* regarding the *Weather* information and so on. Then, we generate 100 transactions for each of interactions based on these parameters. A transaction, as explained in 4.3.1, contains the querying agent (here *A*), the replying agent (either *B* or *C*) and the response with respect to the weather or soil. We further fix the true values of the soil and weather conditions to be **Dry** and **NoRain** so that the decision proposed by the agent should always be **Watering** or “switch ON”. As the responses are based on the assumed trust of the sources (table 6.1) and that we know the true value of the final outcome, we can compute the appropriate rating +1, -1 depending on whether the response was the same as the true value or not. Thus, the simulation step yields a set of transactions of *A* with *B* and *C*.

In the experiment, we assume that at the end of each of the four interactions of the above sequence, the agent *A* validates the information, in order to compute the new values of trust for *B* and *C*. To start the experiment, we consider that all three of the homes have no pre-defined trust associated with them. So, initial trust values of *B* and *C* are equal to 0. With these parameters, we instantiate our MAS to use Eigen trust model and Possibilistic model for the propagation of uncertainty.

#### Results:

The summary of the transactions is shown in the table 6.2. The results of the experiment confirm the fact that most of the time the system predicts the correct decision (61%). Of the remaining 39% of the time when there is a wrong decision predicted, 18% of the time both the sources provide a wrong information. In conflict cases, where the agents provide contradictory information, a correct decision is taken almost twice when compared to the incorrect ones (45 compared to 21).

The trust evolution plot for the two agents *B* and *C*, from *A*’s perspective is shown in figure 6.1.(a). It is evident from the plot that the trust values follow Eigen trust, i.e., at any iteration point the sum of the trust values for all other sources for a specific information type is equal to 1. An iteration is a completion of the sequence of exchanges between *A* and *B*, and then between *A* and *C*, as we described in 6.1.1 above. In the experiment, we generated transactions with initial trust values of *B* and *C* for soil information equal

Table 6.3: Statistics after 100 iterations for almost 50% accuracy of all sources

	Percentage of total suggested decisions	Percentage of decisions suggested without conflict	Percentage of decisions suggested with conflict
<b>Watering</b>	30.00% (30)	26.00% (8)	73.00% (22)
<b>NoWatering</b>	70.00% (70)	62.00%, (44)	37.00% (26)

to 85% and 60% respectively. The trust of  $A$  for  $B_{Soil}$  and  $C_{Soil}$ , initially is 0 as indicated by the pink and blue lines in the figure, and as we had not set any trust for the sources. Thereafter, as  $B$  is assumed more reliable than  $C$  for soil, it provides the true values more often and in accordance with our intuitive opinion the  $\text{trust}(B_{Soil}) > \text{trust}(C_{Soil})$ . Also their final values after 100 iterations are 0.78 and 0.22, which are in accordance with Eigen Trust where the trust measure is computed as  $(+85-15)/(70+20) = 7/9$  and  $(60-40)/(70+20)$ , respectively. We have the same result for the weather information type too;  $\text{trust}(B_{Weather}) > \text{trust}(C_{Weather})$ .

The figure 6.1.(b) presents the uncertainty value of the most certain of the decisions suggested for the agent  $A$ , at the end of each iteration. Since, our rules were based on Possibility theory and that the measures used for uncertainty were in terms of Necessity, we can obtain an estimate for uncertainty as;

$$\text{Uncertainty} = 1 - N \quad (6.1)$$

where,  $N$  is the necessity of maximum of the necessities of all the derived decisions. As seen in the figure, for the initial stages when the trust of the sources is based only on a few interactions, the uncertainty hovers around 0.5. Thereafter, as the trust of the sources increases after a number of interactions, the uncertainty is consistently below 0.5, between 0.2 and 0.3. This supports our hypothesis that information from trustworthy sources leads to less uncertainty in the derived conclusions.

To further prove the hypothesis, we take the trust for the services illustrated in 6.1 to be:  $B - Rain=0.5001$ ,  $B - Soil=0.5002$ ,  $C - Rain=0.5003$  and  $C - Soil=0.5004$ . So, here we effectively consider all the sources to be furnishing the true values with almost 50% accuracy. We use a slight difference, as Possibilistic logic uses min/max logic for computation of necessity values, and hence it poses a problem. The statistics of the generated transactions are shown in the table 6.3. We see that the percentage of correct decisions **Watering** proposed is 30%. In fact, without any experiments, it is obvious that the chances of correct decision ‘ON’ is 1 in 4 = 25%, given the fact that it can only be deduced from **DrySoil** and **NoRain**. This shows without a proper trust management of the sources, an agent will suffer more than in the case where the agent has it.

### **Simplicity and genericity**

With our approach, we envisioned the distributed system as a Multi Agent System. This makes our implementation modular and simpler as the problem of developing a global trust view of the system can be difficult and will suffer the same problem as a centralized trust management paradigm (e.g., single point of failure, bottleneck, performance etc.). Our implementation is mostly modular. We separated the distributed entities into agents; their knowledge base is identified by a common ontology with local beliefs; the rules are written in a standard for semantic web (SWRL) and they contain a third party reasoner (Pellet) to infer new information. An ATMS was first developed and tested as an independent component and then hooked to each of the agents to maintain a consistent local belief base. We assumed the trust of the entities to be managed in terms of ratings. We created a generic abstract class `TrustModel` declaring all abstract methods needed for computation of trust of sources based on ratings. `EigenTrust` is its subclass that we implemented. It uses the ratings of the transactions to compute the trust value of the sources. For any other model of trust, we need to implement the abstract methods in its proper subclass. The uncertainty propagation is restricted to the ontology and the rules. For different models of uncertainty, we need different resolution mechanism introduced into the rules. We implemented Possibility theory resolution clause for this use case. Thus, the implementation is generic. To further strengthen this point, we extend the implementation to include newer improved models of trust and uncertainty for an another application domain in section 6.2.

### **Scalability**

Scalability is the ease with which the implementation can handle the growth of the number of entities and the exchange of messages in the MAS. To measure the scalability of the implementation, we study two aspects.

1. The memory consumption of a single JADE platform as a whole, as we increase the number of pairs of agents (querying and replying) in the system, using the Eigen trust model and Possibility uncertainty theory.
2. For each pair of agents: a querying agent and a replying agent, the variation of the average round trip time for the querying agent to send a message to the replying agent and back and then then computation of trust of the source and uncertainty in the information. We assume the true value of the query is available to the querying agent as soon as it receives response from the replying agent.

The graph of the consumption of memory for different numbers of pairs of querying and replying agents is shown in the figure 6.2. The experiment was

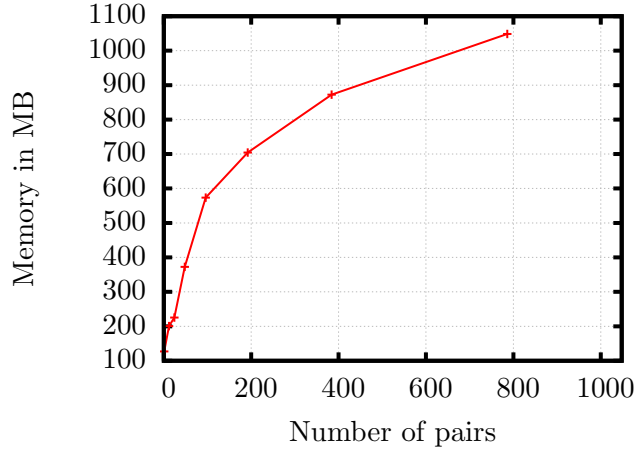


Figure 6.2: Memory usage in a single container

performed on a single JADE container on a DELL Latitude E6420 laptop with Intel®Core(TM) i5-2520M CPU@2.50GHz, 4096 MB of ram, running 64-bit Ubuntu 12.04.5 LTS operating system, java version "1.8.0\_45", Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode) and JADE version 4.3.0. We started with a pair of querying and replying agents and kept the querying and inferring actions of in a `CyclicBehavior`. After each validation, we cleared the belief base and the ATMS contents. The memory footprint of the JADE environment was found to be 126.98 MB. This value increased to 1048.58 MB for 768 pairs of querying and replying agents. The functioning of the querying agent was found as desired. This demonstrates that our agents that consist of several components are stable and the system is scalable in terms of memory. Given the fact that JADE allows multiple containers and multiple hosts, it can be assumed that our implementation can scale horizontally as well.

### Decentralization of trust and uncertainty algorithms

One of our primary goals of the work was to decentralize the trust and uncertainty management algorithms in order to be consistent with the needs of the Intelligent Community type of domains. Hence, our implementation included trust and uncertainty management components in each entity of the distributed system. As such, we were able to implement the Eigen trust model in a decentralized manner for each of the agents of the system. The management of uncertainty in the information, too is specific to agents.

#### 6.1.4 Discussion on the trust and the uncertainty models

As obvious from the experimentation, the Eigen trust is simple, scales well and provides a relative measure for the computation of local trust. However, it does not fare well in certain aspects. First, Eigen trust is relative, and it fails to distinguish between a source which never provided an information and a source that has equal number of good and bad transactions. Also, it does not consider transactions with respect to time. A very good source of information may behave as malicious one for a short period, but still the algorithm considers it as trustworthy because of its past achievements, and vice versa.

We considered Possibility theory for information uncertainty propagation as it is most suited for domains where the information is plagued by incompleteness. The max/min arithmetic of Possibility theory is simple yet powerful. However, we found that in certain situations, this too is inconclusive. For example, a decision has an label  $\{\{A_1, A_2\}, \{A_3, A_4\}\}$ , where  $A_1$ ,  $A_2$ ,  $A_3$  and  $A_4$  are assumptions. Even if we know the necessities of all them, we cannot reason about the necessity of the node as it is not sure which environment derived the node.

## 6.2 Smart City Garbage Collection

This section describes the experimentation and verification steps done for Smart City Garbage collection use case illustrated in 2.4.2. As illustrated, our goal in the work is to apply the trust and uncertainty management models to predict and help in creating an optimal itinerary for garbage collection, thereby optimizing the usage of manpower and resources.

The use case presents an ideal scenario, encompassing the use of citizens and sensors as the sources of information. However, the real world data that we obtained (from Odense Renovations Selskab, The Odense Waste Collection Company, Denmark) as a part of application of our trust and uncertainty management models and algorithms is dissimilar in a handful of ways to the use case that we portrayed. We first explain, the data set that we obtained from the garbage company in section 6.2.1 and then present the dissimilarities and enhancements needed to the data set in the section 6.2.2 respectively.

### 6.2.1 Analysis of the Data set

The data set contains an elaborate modeling of different aspects of garbage collection for the city of Odense. The garbage collection company of Odense, uses garbage trucks called *Units* for collecting garbage present in different parts of the city. The citizens generally store the garbage in containers called “bins”, present either outside their premises or in some cases within

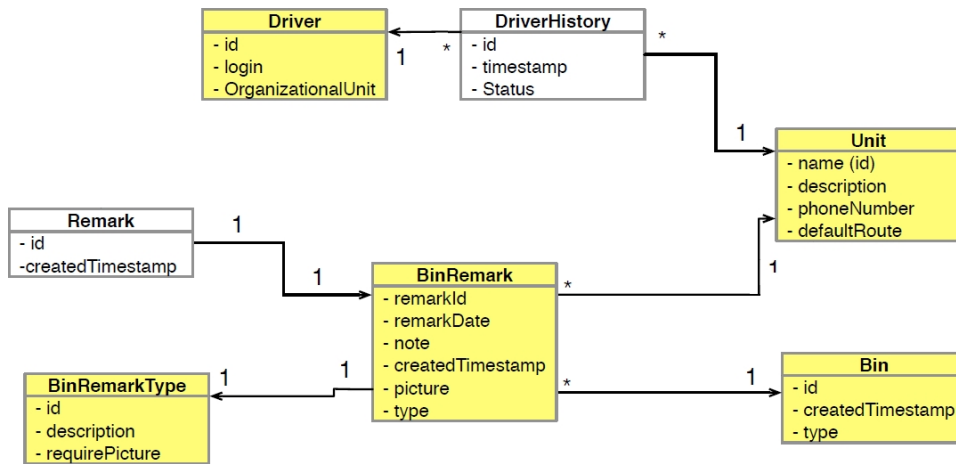


Figure 6.3: The Smart city garbage collection data model (Source: A2I Systems)

the households. The bins are of different types, e.g., some bins store only paper while others store everything. The company deploys drivers and other helpers for collecting the garbage from the bins. They perform different tasks such as emptying the containers, verifying the conditions of the bins, dredging the bins etc. Depending upon the tasks that they perform, they provide a remark for the bin that is stored in the system in real time. The garbage is then carried to the disposal site for disposing off. The itinerary for garbage collection found in the data set seemed static, i.e., there were garbage collection itinerary every week or every fortnight, or in some cases every month.

The size of the entire data set in zip format is about 25 GB. Of the different aspects of garbage data present in the data set, we need to model the possible sources of information, the information about garbage in a particular location informed by sources, the rules that can be applied to infer higher level information. Hence, we only require a subset of the entire data set for the application of our models. We present the necessary subset of the data set in the figure 6.3. A small description of the various tables in the figure above are explained below.

1. Driver: This table stores the data pertaining to different drivers, their ids and the organizational unit that they belong to.
2. DriverHistory: This table stores different events related to a driver, such as: when did a particular driver sign into the system or signed off the system.
3. Unit: This table represents a means of transport used for collection the collection of garbage.

4. Bin: This table stores the information regarding the garbage containers present in the city. The type of bin and its location are stored in two other tables `BinType` and `BinLocation` respectively.
5. BinRemark: This table stores the remarks given by drivers about a bin. The bin remarks can be of several types that are stored in the table `BinRemarkType`.

### 6.2.2 Dissimilarities of the data set to our initial use case

In this section, we present the dissimilarities of the available data set with respect to our idealized use case in section 2.4.2. We conceived the idealized use case with the help of on going works in the project FIWARE and discussions with other research teams in Orange Labs working in diverse domains. We later came in contact with Odense Garbage Management (OGM) company, who were kind enough to share their data set with us. Though the basic intent of garbage collection in the use case conceived by us and that of OGM are same, there exist a handful of dissimilarities. They are listed below.

1. The idealized use case assumes the presence of sensors in the garbage containers and the citizens of the city acting as possible sources of information. However, the city of Odense has not yet implemented such mechanisms. The only sources of information in the data set are the drivers, who provide remarks about the conditions of the bins after collection of garbage.
2. In our use case, we hypothesize that the values of uncertainty in the information and the trust values of the various sources will help in planning an ideal itinerary for garbage collection. But, with the real world data set, we have the data of the garbage collected in the city in the past year. Hence, we can only contemplate on computing trust values of sources from the available data set and assigning uncertainty measures for different high level information that can be derived from there. But, we have no method to confirm it ourselves in true sense.

### 6.2.3 Objectives of this application

We have some specific objectives with regards to this work in addition to those that we expressed in the beginning of the thesis.

1. Propose a trust model that fits the data of the data set.
2. Analyze different remarks about a bin, and based on the remarks, reason a possible course of action to be taken. In other words, discover the rules for the domain.



id	unit	driverLogin	timestamp	status
35144	11015	PEP	1342501038206	LOGIN
35145	11120	SBF	1342501076179	LOGIN
35148	11003	BOP	1342501249214	LOGOUT

Table 6.4: A snippet of `DriverHistory` table

3. Apply our approach, i.e., assigning uncertainty to a remark based on the trust of the driver who provided the remark, and then infer higher level information with corresponding uncertainty values.
4. Analyze the approach against the initial success criteria.

#### 6.2.4 Enhancements and assumptions in the data set

The database schema in the data set seems to have been built with the ease of storing the garbage collection information. In order to use the data set, we needed to make the following enhancements and assumptions.

1. It is not evident from the data set about who the driver of a particular unit is at a particular instance of time in the data set as illustrated in the table 6.4. The table stores when the driver has logged into the system and when the driver has logged out of the system and what Units they were driving. So, to know about who the driver of a unit was at an instance of time, we need to search for the driver who logged in just prior to and just after, this time instance with the particular unit. Further, the table was found to contain quite a few occurrences of the cases where driver logged in, but never logged out and vice versa. All such instances were cleaned from the table with the assumption that they are faulty.

We need this information because it is a part of “*A source says some information about something at some time instance*” statement used in our approach. The remarks about the bins (`BinRemark`), on the other hand, are related to the `DriverHistory` table via the `Unit` table and the linking attributes are the unit name and the remark created timestamp.

2. After analysis of the data, we found that the drivers leave a remark about a bin after visiting and inspecting its location. Also, drivers do this visit in a random manner: sometimes once in a week and sometimes once in a couple of days and so on. This may be based on calls from citizens of a particular locality. However, this information is missing from the data set. Hence, to illustrate the fact that an information can arrive from multiple sources, we ignore the time stamp

Driver	Number of remarks	Trust
PEP	23	0.46
OVR	10	0.2
STA	5	0.1
LOP	12	0.24

Table 6.5: Trust computation based on number of remarks

in the information and take a block of one month period and assume all remarks within this period for decision making. This way we have multiple drivers visiting a bin location and providing different remarks, thereby constituting multiple sources of information.

### 6.2.5 Simplified trust model

In our previous use case 6.1, we use Eigen trust as the trust model, as we have different entities of the system that need to compute trust for themselves. In Smart City Garbage Collection use case, however, we have a central administrative authority that is responsible for planning the itineraries of different drivers and vehicles. The sources of information for this system are the drivers who provide remarks about the various garbage bins. Furthermore, there is no way of validating the remarks provided by the drivers. Hence, the previous trust model does not fit this use case. As a result, we need to model trust with respect to some other parameter available in the data. From the analysis of the available data, we found that a possible method to quantify trust of the drivers, who are the only information sources in the use case, is the number of remarks that were provided by a driver, i.e., the more the number of remarks, the more trustworthy he/she is. This is justifiable as all drivers are known to the garbage company beforehand and the more they provide comments means that they are more used to providing remarks than others, and hence more trustworthy. An example of this is shown in the table 6.5. We compute trust as a weighted sum of the number of remarks provided by the driver. If we have  $n$  drivers in our data set the trust of  $i^{th}$  driver can be computed as:

$$T_i = \frac{\sum_{Driver=i} remarks}{\sum_{Driver=0}^{n-1} remarks} \quad (6.2)$$

### 6.2.6 Uncertainty model

As discussed in our approach in 4.6, we assign uncertainty to an incoming information based upon the trust levels of the provider of the information.

We then use rule based inferencing to obtain different conclusions. The propagation of uncertainty from the lower level information (e.g. the incoming information) to the higher level information (the conclusions) is done using a model of uncertainty. In the Intelligent Community use case, we used Possibilistic model for uncertainty propagation. As we pointed out in 6.1.4, it is inconclusive in cases where an ATMS node has two or more environments. So, we chose Probabilistic model as it is a widely accepted theory. Other motivation for the use of Probabilistic model for uncertainty propagation in this case is that we know all the remarks and the conclusions that can be derived from these remarks beforehand. Conditional probability, which measures the probability of an event given some other event has occurred, fits perfectly to such scenario.

For this use case, some of the higher levels of conclusions that can be derived from the remarks of the drivers are illustrated in the figure 6.4. The boxes at the bottom of the figure are the conclusions and the information in the boxes above are the possible remarks of the drivers. E.g., remarks like "Defective container" and "block for bin" indicate a bin missing event, so we can infer that the bin needs a replacement of a new installation (Replace/Install new bin).

### 6.2.7 Experimentation

In this section, we explain the details of how we obtained the subset of the real world data set for experimentation and what experiments we carried out with the data set.

The data set is a Microsoft SQL Server based database and contains several aspects of garbage collection of the city of Odense. We joined several tables to obtain the related information: *Who are the drivers that provided remarks about different bins in the city?* There are around 2038415 remarks about the different bins of the city from 2011-03-28 to 2015-01-14. As explained in section 6.2.4, we ideally need multiple sources of information for an event in our problem description. However, as this is not available in the data set, we assume that all remarks about a bin, available during a period of time (say a month), were available to us at once. This way we have multiple remarks about a bin, possibly from different drivers. Also, as not all the bins are visited equally regularly, we do not have same number of remarks for all bins. So, to make the test uniform, we consider those bins that are visited most by the drivers. For the first experimentation, we considered the top 10 bins that were visited most by the drivers and the corresponding remarks given for the bin during the visit. We found that there were in total 1772 remarks of these bins and the corresponding numbers for each of the remarks is shown in the table 6.6. The various conclusions that can be inferred from the remarks and their corresponding numbers in the data set are shown in the table 6.7. From the table, it can be seen that

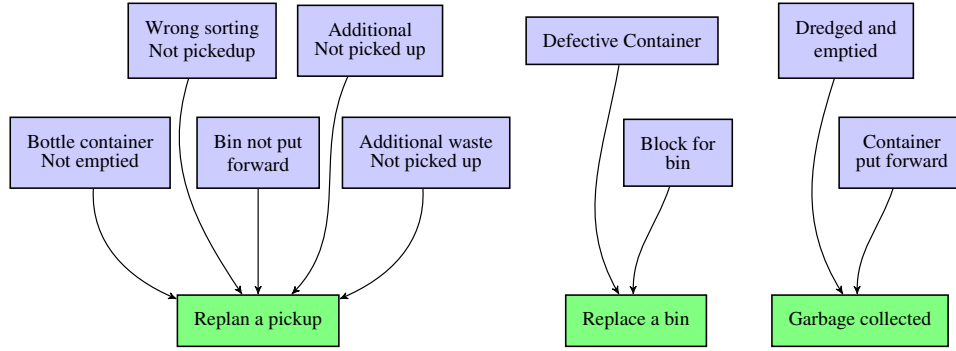


Figure 6.4: Some remarks and the conclusions that can be derived from them

majority of remarks infer *Replan*, while a few others infer *Replace Bin* and *Garbage collected*. Using this table and the probability theory explained in 4.4.2, we obtain different conditional probabilities needed for obtaining the probability of the conclusions. The computed values of the probabilities are as shown below. We use these values to obtain the probabilities of the conclusion using the equation 4.6.

$$P(\text{NotPutForward} \mid \text{Replan}) = 1$$

$$P(\text{Thecontainersetforth} \mid \text{Replan}) = 0$$

$$P(\text{Defective} \mid \text{Replan}) = 0$$

$$P(\text{Dredged} \mid \text{Replan}) = 0$$

$$P(\text{Suspended} \mid \text{Replan}) = 0$$

$$P(\text{NotPutForward} \mid \text{Replace}) = 0$$

$$P(\text{TheContainersetforward} \mid \text{Replace}) = 0$$

$$P(\text{Defective} \mid \text{Replace}) = 0.6$$

$$P(\text{Dredged} \mid \text{Replace}) = 0$$

$$P(\text{Suspended} \mid \text{Replace}) = 0.4$$

$$P(\text{NotPutForward} \mid \text{GC}) = 0$$

$$P(\text{TheContainersetforward} \mid \text{GC}) = 0.6$$

$$P(\text{Defective} \mid \text{GC}) = 0$$

$$P(\text{Dredged} \mid \text{GC}) = 0.4$$

$$P(\text{Suspended} \mid \text{GC}) = 0$$

If we consider a subset of the 1772 remarks for each iteration. Here, an iteration is where a number of drivers would ideally provide the remarks about a particular bin before the system can plan a itinerary. As the data

<b>Remark</b>	<b>Count</b>
Not put forward	1762
The container set forth (the reason)	3
Defective container (triggers an exchange)	3
Exhumed and emptied	2
Suspended for container	2

Table 6.6: Remarks and their counts

	<b>Not put forward</b>	<b>The container set forward</b>	<b>Defective</b>	<b>Dredged &amp; Emptied</b>	<b>Suspended</b>
<b>Replan</b>	1762	0	0	0	0
<b>Replace Bin</b>	0	0	3	0	2
<b>Garbage Collected</b>	0	3	0	2	0

Table 6.7: Remarks and different inferences

does not seem to be compliant with this, we assume the subset. Let for instance we consider ten remarks to form an iteration. Then, of the ten remarks, we can have ten different or same conclusions. The conclusions follow one of the three cases explained in 4.8 and we resolve the conflict by taking those conclusions which are most certain. Based on this, the system can plan an effective itinerary. The data set we presented here is skewed for the fact that majority of the remarks for the bins are “Not put forward”. Hence, for this case, most of the conclusions that is deduced is “Replan”.

In the next section, we present the verification of the experiments that we performed here.

### 6.2.8 Verification of the use case

Like the verification of the Intelligent Community use case, we verify this use case against in two points. Firstly, we present the validation of the hypothesis in section 6.2.8 and validation if our success criteria in section 6.2.8.

#### Verification of the hypothesis

Our hypothesis that trustworthy sources provide more certain information and vice versa cannot be verified in the given data set as we lack information regarding whether the reported remark was found true or false. The data set contains only the remarks about different bins by the drivers at different instants of time. But it does not contain information whether the remarks were later confirmed.

### Verification of success criteria

1. **Simplicity and genericity:** The extension of our proposed solution to fit the Smart City Garbage Collection use case was: to develop a domain specific ontology using the generic ontology, to define the subclasses and instances of `Information` and `InformationSource` and to declare the domain specific rules for inferring higher level information. The extension involved creation of a trust model specific to the data of the smart city, computing the various conditional probabilities of conclusions given the remarks of the drivers and probabilistic rules to propagate uncertainty. Though the trust and uncertainty models we used were simple, we had to alter them compared to what we used in the first application. This was to suit the application domain and the nature of available data.
2. **Scalability:** The data set we used for the experimentation had 80 distinct information providers (drivers) who provided over 200,000 remarks about bins in different localities of the city. The conditional probability values about the remarks leading to various conclusions were computed beforehand and later used by the agents to compute their certainties. For the given data set, the models were found to be scalable.
3. **Decentralization of trust algorithms:** The Smart City garbage collection is distinct, in the sense, the objective of the use case is managing trust of the remark providers by a central authority. As such we did not need a decentralization here.

## 6.3 Project FIWARE

FIWARE<sup>3</sup> is a European project (Project No. 285248) that comes under the umbrella of Future Internet - Public Private Partnership (FI-PPP<sup>4</sup>). The goal of the project is to build the Core Platform of the Future Internet. The project is aimed at building modular components that are reusable in a number of different projects. They are called the *Generic Enablers* (GEs). This way, different projects can reuse the enablers to achieve the desired qualities. The precise definition of a GE is below.

A functional building block of FIWARE. Any implementation of a FIWARE GE is made up of a set of components which together supports a concrete set of Functions and provides a concrete set of APIs and interoperable interfaces that are in compliance with open specifications published for that GE.

---

<sup>3</sup><https://www.fiware.org>

<sup>4</sup><https://www.fi-ppp.eu>

As we explained in the section 1, and as presented in the work [MBF<sup>+</sup>12], management of trust and uncertainty is an essential need of various projects under FIWARE. Hence, through this work, we intend to create a GE for resolving the issues of management of trust and uncertainty.

### 6.3.1 Introduction

Context awareness is considered as a key technology within the IT industry, for its potential to provide a significant competitive advantage to services providers and to give substantial differentiation among existing services. According to a Gartner Inc. report [Lap09], “Context-aware computing today stands where search engines and the web did in 1990”.

In parallel to this, the interest of the scientific community in the context aware computing domain has gained a lot of momentum, due to the fact that with the advent of the Internet of Thing (IoT) era, terabytes of data are bound to be produced daily by sensors and equipments.

Such data, when correctly interpreted can enrich the description of the context, which in turn makes it possible for services and applications to become context-aware, and finally to improve their efficiency in terms of personalization, and simplicity of use.

However, inherently to the multiplicity of data sources, it becomes necessary to evaluate the quality of data. This issue arises because some sources could be unreliable or untruthful. This is particularly true in domains where adversity exists. For example, in intelligence warfare where false information might be distilled by enemies to create confusion. This is also true in today economic life, where due to intense competition, incorrect information or hoaxes could be anonymously broadcasted by third parties to discredit competitors. In our IoT domain, different environment sensors might be unequally reliable, simply because some might occasionally malfunction. In such cases they might deliver measurements in which we have different level of confidence. Thus, we need a mechanism to quantify the trustworthiness of the sources and manage its evolution over time. For this reason, we have extended a IoT context broker [RGMSE14], with specific components to handle trust and uncertainty management.

### 6.3.2 Home mood Scenario

People do care about their home, where they spend more than half of their lifetime [atu]. When they are away from home they are concerned by the comfort and protection of their family who are still there or by keeping their home safe from burglary and any natural disaster if the home is empty. Homes takes a important place in people’s heart and people has strong affective links with their homes. Recent studies have investigated the use of social network media such as facebook, twitter to establish a privileged

connection between people and their home, like they do for their friends. Thus, homes could chat, tweet and send selfies to keep people reassured about their home and its occupants *find references*. In our work, we investigate mechanisms and strategies to infer high level concepts such “the home is empty”, “most people are sleeping”, “the home is being broken into” from low level data such as “move detection”, “temperature” and any data produced by home automation devices. We call these high level concepts “moods”, because they synthesize both a general state of the home to which people generally associate positive or negative emotions (such as “fear of intrusion”, “satisfaction of a quiet and peaceful atmosphere”)

In this preliminary work we address the problem of inferring information about presence in the room based on raw data produced by the homelive sensors deployed in the room. More specifically, our Home friend tells us about the number of people it hosts. The possible outcome are enumerated as the following:

- nobody’s there
- there’s at least one person here
- there’s at least two persons here

This information is displayed on a virtual wall such as that of the status area of a facebook page, or could be provided by the Home friend if we send it a text message asking “is somebody’s in?”

In the following section, we introduce the FIWARE Linked Open Data (FLOD) enabler and its extension for managing trust and uncertainty.

### 6.3.3 Approach and architecture

The task of inferring high level information from low level data has been investigated in the field of Artificial Intelligence and is still today an active research topic. Iconic instances of that problem solving task include diagnosis in the medical domain, where an disease has to be identified based on some observed symptoms and image understanding, where a person has to be identified from an still image, or an human activity has to be recognized from a video sequence.

There are alternative approaches for designing solutions to this type or problem solving task. To the one that is using neural networks (NN), and its popular variant called “deep learning”, we have preferred a symbolic approach where inferences are based on domain expertise rules that are triggered by a dedicated rule engine. We have made this choice because the NN approach requires a preliminary stage of supervised learning, which itself requires a lot of sensor data to be collected, stored and properly formatted and manually labeled with the different interpretation classes to be inferred.



In our previous work we have developed a context broker enabler called FLOD which is dedicated to collect context data from any data source and publish it to any application that could use this context data for adapting its behavior to the current context. One salient feature of FLOD is that it models context information using RDF which makes it compliant to a W3C standard and makes possible the use of domain ontologies that constrains RDF primitives to denote instances of classes or relations defined in the ontologies. This additional feature ensures FLOD semantic interoperability with its potential client applications and other Linked Open Data sources. Semantic interoperability means that context data provided by FLOD to the client application will be understood and correctly interpreted by the client without requiring the client to agree with the data source about what the data means, how the data is represented and structured, and where to find the bits of interest to the client. Conversely, the data source will be able to produce its data without anticipating or conjecturing ways this data will be interpreted and used by the prospective client or data consumer.

Semantic Context Management Service (CMS) [RPS<sup>+</sup>07]. This has been adapted to the IoT domain [FIW].

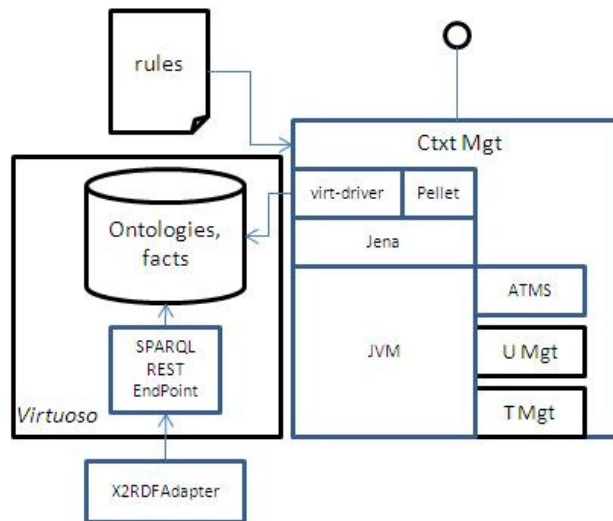


Figure 6.5: FLOD IoT Semantic Context Broker

As prescribed by cognitive engineering methodologies and good practices in the domain, we have looked up existing ontologies that cover our universe of discourse, i.e. the scope of concepts and issues that we need to model in our application. We ended up using the [IoT] and [Unc] ontologies to model IoT devices and data measurements and to model data uncertainty management. We didn't find any ontology for modeling trust management. For this reason we developed our own trust ontology. In the figure 5.3,

we depict the main classes from these three ontologies together with their interrelationships.

In the next section, we illustrate how our extended enabler has been used to implement a system which solves the scenario described in section 6.3.2.

### 6.3.4 Scenario implementation and experimentation

In this section, we describe the system in terms of functionality and software architecture and main components. The architecture of the FLOD context broker and our generic enabler is shown in the figure 6.5. It depicts the implementation specific details of the enabler. The events from sensors of a deployed scene are stored in a RDF triple store database. The data and the sources of data are represented in terms of classes and individuals of the ontology that we explained in 5.2. The enabler runs on top of JVM. The trust and uncertainty management components (UMgmt and TMgmt), and the ATMS are developed in Java. The `virt-driver` enables loading and manipulation of the linked data as a JENA model. The domain specific rules are input to the enabler as an external resource. Based on them and the available triples about the domain, the pellet reasoner is then able to load and execute the rules and infer new information. The explanations for various inferred information are then fed to the ATMS to obtain the uncertainties of the conclusions.

#### Sensors trust

In our scenario, there are no malicious data sources. All sensors have been deployed by a reliable party, most of them have been indeed installed by the home owner himself. Besides, we have checked that under nominal operational state each sensor provide the correct measurement. The three cases where the sensors provides outlying measurements are when they are in defect state (mechanically or electronically damaged), when they lack electric power supply (too low battery level) and finally when there's a radio connection loss. In our installation, there is a device management facility that keep track of the battery level of each device and another that detects sensor measurements outliers. For these reason, in our experimentation, we assume that the only component of sensors reliability, or trust is the quality of radio communication. Thus the trust we assign to each device is proportional to how much the sensor is connected to the base. If in average, a sensor experiences connection losses during a cumulated time  $\delta_{cl}$  for a cumulated time  $\delta_{cc}$  of correct connection, its reliability is:

$$Reliability = \frac{\delta_{cc}}{\delta_{cl} + \delta_{cc}} \quad (6.3)$$

This reliability measure is the trust that our system assign to the sensor.

### Presence inference

In our experimental setting, there is a unique entry door to access the room. This door has an automatic door closer. This configuration enables us to conjecture that if a sign of activity is detected in the room after the door has been closed we can infer that there's somebody in the room. There are several such signs including, the opening or closing of a drawer or one of the two fridge door, the detection of movement reported by any of the 5 motion detectors and the switching on or off of the TV or of the boiler. The switching on/off of the coffee machine cannot be taken into account because it is a programmable device which can switch on/off on its own through programming.

We model this reasoning by defining a rule that infer a sign of activity for each possible cause. Then we have a rule that infer presence from a comparison between the last activity reported and the last entry door closing. As explained earlier if the last activity is anterior to the entry door closing, the conclusion part of the rule specifies that there's someone in the room. We show one of the activity reporting as well as the presence inference rules below:

```
says(?dev1, ?info1)
hasName(?dev1, "MLMove3"),
hasValue(?info1, MoveDetected),
timestamp(?info1, ?move3ts),
says(?dev2, ?info2)
hasName(?dev2, "MLDoor1"),
hasValue(?info2, DoorClosed),
timestamp(?info2, ?door1ts),
greaterThan(?move3ts, ?door1ts)
->
believes(MySelf, SomebodyIsInside)
```

This rule tells that if a door sensor and a motion sensor in a room detect events in a close interval of time, we can infer that there is somebody in the room. In order to infer the presence of more than one person, we analyze the simultaneity of activities occurring in different places. For instance if one fridge door is open at the same time than a move is detected in the living room area, one single person cannot be responsible for these two activities. We can conclude that there are at least two person in the room. We show one of the activities simultaneity detection rule below:

```
says(?dev1, ?info1)
hasName(?dev1, "MLMove3"),
hasValue(?info1, MoveDetected),
timestamp(?info1, ?move3ts),
```

```
says(?dev2, ?info2)
hasName(?dev2, "MLMove4"),
hasValue(?info2, MoveDetected),
timestamp(?info2, ?move4ts),
lessThan(abs(subtract(?move3ts, ?move4ts)), 500)
->
believes(MySelf, MoreThanOneInside)
```

Here, two motion detectors `MLMove3` and `MLMove4` deployed relatively far from each other in the room, detect motions within 500 milliseconds. From this, we can infer that there are more than two moving objects and/or people. Once the presence of one, or two or more people is inferred this presence information persists until the entry door is open again. Once the door is opened, this presence information has to be removed because the people inside could have exited the room.

### Assessing presence information reliability

The trust level of sensors referred in rules (reference) and measured as explained in the paragraph (reference) is converted into uncertainty of the sensor measurement. The ATMS is then used to propagate these uncertainties into the activity detection conclusion and the presence information inference if this rule is triggered. The uncertainty of presence then represent the confidence we have that someone or more than two people are in the room. If the rules are not triggered, it doesn't mean that nobody in the room. However, if no activity has been detected for a certain time, we could hypothesize that there are nobody in the room.

We have experimentally tested our system by asking people to come and go in the room and by submitting queries to the system about presence information. The results are correct, if the time of the query is far enough from the last time the door has been closed, which indeed is an expected artifact. This promising results has yet to be consolidated through extensive experimentation campaign.



## Chapter 7

# Conclusion and Discussion

This chapter presents the conclusions and discussions of this thesis. We explain the achievements and contributions of our work in section 7.1. The limitations and the future enhancements of the work are described in 7.2 and finally we presents the perspectives from this work in 7.3 where we explain how the approach can be extended beyond the use cases of this work.

Today, distributed systems and many other domains that deal with management of information from multiple sources face a challenge to classify and select the best sources of information. Quantification of uncertainty in the available information is yet another challenge. We assumed that trust measure of a source can be an indicator the quality of information provided by it. Thus, our approach was to put them together to infer high level information in an application domain was our approach. We applied this to two distinct application domains: *Intelligent Community* and *Smart City Garbage Collection*. Different experiments and results supported our hypothesis for these applications. As such, our major contributions and achievements of this thesis are as follows.

### 7.1 Contributions and Achievements

The following section illustrates our achievements from the thesis.

1. Verification of the hypothesis: Our hypothesis was “In the absence of any other information, the data uncertainty is inversely proportional to the trust on the data source that furnished it”. From the experiments in section 6.1.3, where we use Eigen Trust and Possibility theory as trust and uncertainty models respectively, we see that as the trust values of the sources increase over several iterations, the uncertainty of the decision gets lowered. Also, the results in tables 6.1 and 6.3 show us that the system would have faltered far more in the absence of trust and uncertainty values for decision making, which is in strong support of our hypothesis.

2. Study of two trust and uncertainty models: As a part of the work, we explored several trust models as discussed in the state of the art and having analyzed them, we found two trust models most suited for our application domains, namely the Eigen Trust and the  $\beta$ -reputation. As the Smart City data set lacked the information about the quality of remarks (good or bad), we could not apply  $\beta$ -reputation. We used a simplified trust model based on the number of remarks as the basis for modeling the trust. Similarly, we analyzed the use of Possibility theory and Probability theory for inferring high level information in the two applications respectively.
3. Linked data and SWRL for modeling information and domain rules: We used ontologies for representation of the knowledge of the agents. The generic ontology contains the definitions of the sources of information, the models of trust and uncertainty and the relations between how an information is transformed into beliefs of the agents and then into the inferred conclusions. The ontologies of the application domain extend the generic ontology to represent the domain specific knowledge. The rules to infer new information from the existing ones are encoded in SWRL format in the domain ontology itself. Hence, extending our approach to a new application domain is straightforward.
4. Agents with reasoning capabilities: We proposed and implemented agents of the system augmented with reasoning capabilities. Each of the agents contain a reasoner, a trust module, an ATMS and various behaviors. The results have shown that such an amalgamation of components is scalable, simple and generic to multiple application domains. The domain intelligence is inserted into an agent via the domain rules in the ontology.

Thus, in this thesis we illustrated the challenges of incomplete and uncertain information provided by sources of information and the problems faced by the current distributed systems that do not the use of the trust measures of the sources of information. In particular, modeling and quantification of the uncertainty in the information and its propagation while inferring high level information is utmost important in order to know the enrich the conclusions with uncertainty. We used possibilistic and probabilistic models of uncertainty for the use cases that we illustrated. For trust management, we employed the EIGEN trust and a simplified trust models for the computation of local trust for the two domains of applications. We put forward various components needed by an agent to manage trust for other agents in the network and then reason about uncertainty in the derived or inferred information. We evaluated the functioning of agents against simulated data and real data from a European smart city. We extended our work further to use the algorithms as a generic enabler for other projects.

This section presented our contributions and achievements towards the thesis and in the following section, we elucidate the limitations and future enhancements of the thesis.

## 7.2 Limitations and future enhancements

This section describes the limitations and future enhancements of the work. We list them as follows:

1. The main aim of the thesis was to develop a framework that could support our hypothesis. The study and comparison of different existing trust and uncertainty models is not our intention; though we selected the best fitting models for the application domains based on their requirements. We intend to study more models of trust and uncertainty as future enhancements.
2. In our work we have so far ignored the transactions related to the cases described in 4.8. This is because we do not have sufficient information to confirm which of the sources of information are either true or false (depending on the cases). This may, many a time, form a considerable chunk of transactions. A possible solution is to validate individual environments that form the label of such derived nodes. We intend to address these issues in our future work.
3. We applied our work to Intelligent Community use case which is a real time hypothetical system. Hence, we had to verify the system against simulated data. For, the Smart City use case on the other hand, we had the data from the past year. Due to the lack of information, the verification of this data was not possible. Hence, our approach needs a verification against real time system with a real use case.
4. An interesting theory for combining evidences from multiple sources and arriving at a conclusion with a degree of belief is *Dempster Shafer Theory* (DST). It is based on theory of evidences. It would be an exciting prospect to use this theory along with our approach. The only drawback with DST is that in extreme cases, it may produce counter intuitive results.

## 7.3 Extending beyond the explained applications

The connected world has evolved greatly over the past decade in terms of the number and types of devices, the underlying technology and the use cases to simplify the daily life of the humans. Sharing of information has become a quintessential part of most of the present day applications and it continues



to grow as newer devices and domains are invented. For all distributed domains, that involve sharing of information most certainly need a notion of demarcating different sources of information based on their trust and reason on that information. This is where our thesis can become a potential solution. As such we contemplate of two interesting applications below.

1. Cloud computing: It can be broadly classified into three service models.
  - (a) Infrastructure as a Service (IaaS). e.g. RackSpace, Amazon, etc.
  - (b) Platform as a Service (PaaS). e.g. Heroku, Google AppEngine etc.
  - (c) Software as a Service (SaaS). e.g. GMail, Google Drive etc.

All these service models primarily involve two parties: the cloud providers (CP) and the cloud consumers (CC). Some researches [HHRM12] also include Cloud Brokers (CB), which do not provide the cloud services directly nor are real cloud consumers but act as an intermediary between the CB and the CP. Many CPs provide all three types of services with further sub categorization in terms of the details of the services. E.g., Google app engine can provide services for specific environments like: Python, Java etc. Thus, with Business-to-Business (B2B) and Business-to-Consumer (B2C) business models in cloud computing, the need for a consumer is to rate and select the best service provider depending on different criteria. Some important ones' are trust, data privacy and security. Apart from these, a consumer also takes into account factors like QoS (measured in terms of response time, availability and elasticity), the price, various Service Level Agreements (SLAs), third-party reputation or recommendation etc., before making a selection of a provider. All these criteria can form the basis of quantification of trust of the CP. Thus, cloud computing can be an exciting application for our work.

2. Big data analysis: A major issue with the *Big data* applications is improving quality of information (QoI). As we explained in the introduction, QoI is measured in terms of different parameters; two of the most important ones' are trust of the source of information and the uncertainty in the information. A possible extension of our work can be enriching big data applications with trust and uncertainty values thereby improving the QoI of the data.

# Bibliography

- [ABB<sup>+</sup>08] Hidir Aras, Clemens Beckstein, Sonja Buchegger, Peter Dittich, Friederike Klan, Birgitta Knig-Ries, and Ouri Wolfson. Uncertainty and Trust. In *Proceedings of the Dagstuhl Invitational Seminar 08421*, pages 1–3, 2008.
- [Abe01] Karl Aberer. P-grid: A self-organizing access structure for p2p information systems. In *Cooperative Information Systems*, pages 179–194. Springer Berlin Heidelberg, 2001.
- [AD01] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the Tenth International Conference on Information and Knowledge Management, CIKM '01*, pages 310–317, New York, NY, USA, 2001. ACM.
- [ARH97] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the 1997 Workshop on New Security Paradigms, NSPW '97*, pages 48–60, New York, USA, 1997. ACM.
- [atu] American time use survey. <http://www.bls.gov/tus/charts/>. Accessed: 2015-11-15.
- [BG10] Amandine Bellenger and Sylvain Gatepaille. Uncertainty in ontologies: Dempster-Shafer theory for data fusion applications. *Workshop on Theory of Belief Functions*, 2010.
- [CCB<sup>+</sup>06] MJ Carey, S Ceri, P Bernstein, U Dayal, C Faloutsos, JC Freytag, G Gardarin, W Jonker, V Krishnamurthy, MA Neimat, et al. Data-centric systems and applications. 2006.
- [CDBN11] Andrea Caragliu, Chiara Del Bo, and Peter Nijkamp. Smart cities in europe. *Journal of urban technology*, 18(2):65–82, 2011.

- [CF01] Cristiano Castelfranchi and Rino Falcone. Social Trust: A Cognitive Approach. *Trust and Deception in Virtual Societies*, pages 55–90, 2001.
- [CG03] Vinny Cahill and Elizabeth Gray. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2003.
- [DK86] Johan De Kleer. An assumption-based TMS. *Artificial intelligence*, 28(2):127–162, 1986.
- [DLG<sup>+</sup>04] Mark D’Inverno, Michael Luck, Michael Georgeff, David Kinny, and Michael Wooldridge. The dMARS Architecture: A Specification of the Distributed Multi-Agent Reasoning System. *Autonomous Agents and Multi-Agent Systems*, 9(1/2):5–53, 2004.
- [EY09] Amira Essaid and BB Yaghlane. BeliefOWL: An Evidential Representation in OWL Ontology. *Proceedings of the Fifth International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2009)*, pages 1–4, 2009.
- [FIW] The project FIWARE. (Project No. 285248). <https://www.fiware.org>. Accessed: 2015-11-15.
- [GI90] Michael P Georgeff and Francois Felix Ingrand. Real-time reasoning: The monitoring and control of spacecraft systems. In *Artificial Intelligence Applications, 1990., Sixth Conference on*, pages 198–204. IEEE, 1990.
- [GPP<sup>+</sup>98] Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Wooldridge. Intelligent Agents V Springer-Verlag Lecture Notes in AI. In *Intelligent Agents V: Agents Theories, Architectures, and Languages*, pages 1–10. Springer, 1998.
- [GW11] Virgil Gligor and Jeannette M Wing. Towards a Theory of Trust in Networks of Humans and Computers Humans and Computers. *19th International Workshop on Security Protocols*, 2011.
- [GYL12] Xi Gong, Ting Yu, and Adam J. Lee. Bounding trust in reputation systems with incomplete information. *Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY ’12*, page 125, 2012.
- [Hal98] Joseph Y Halpern. *A logical approach to reasoning about uncertainty: a tutorial*. Springer, 1998.

- [Hec92] David Heckerman. The certainty-factor model. *Encyclopedia of Artificial Intelligence*, pages 131–138, 1992.
- [HG12] Christina Hochleitner and Cornelia Graf. Making Devices Trustworthy: Security and Trust Feedback in the Internet of Things. *Pervasive'12 Fourth International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use*, 2012.
- [HHRM12] Sheikh Mahbub Habib, Sascha Hauke, Sebastian Ries, and Max Mühlhäuser. Trust as a facilitator in cloud computing: a survey. *Journal of Cloud Computing: Advances, Systems and Applications*, 1:33, August 2012. Provisional version.
- [Hur06] Robert F. Hurley. The Decision to Trust. *Harvard Business Review*, 2006.
- [HWS08] Chung-Wei Hang, Yonghong Wang, and Munindar P Singh. An adaptive probabilistic trust model and its evaluation. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1485–1488. International Foundation for Autonomous Agents and Multiagent Systems, 2008.
- [IoT] Internet of Things Architecture (IoT-A) Deliverable D1.5 Final architectural reference model for the IoT v3.0. [www.iot-a.eu/public/public-documents/d1.5/at\\_download/file](http://www.iot-a.eu/public/public-documents/d1.5/at_download/file). Accessed: 2015-11-15.
- [JBXC08] Audun Jøsang, Touhid Bhuiyan, Yue Xu, and Clive Cox. Combining trust and reputation management for web-based services. In *Trust, Privacy and Security in Digital Business*, pages 90–99. Springer, 2008.
- [JI02] Audun Jøsang and Roslan Ismail. The Beta Reputation System. In *Proceedings of the 15th Bled Electronic Commerce Conference*, volume 5, pages 2502–2511, 2002.
- [JMP06] Audun Jøsang, Stephen Marsh, and Simon Pope. Exploring Different Types of Trust Propagation. In *Proceedings of the 4th International Conference on Trust Management, iTrust'06*, pages 179–192, Berlin, Heidelberg, 2006. Springer-Verlag.
- [Jøs96] Audun Jøsang. The right type of trust for distributed systems. In *Proceedings of the 1996 workshop on New security paradigms*, pages 119–131. ACM, 1996.

- [KKRMvK09] Christoph Koch, Birgitta König-Ries, Volker Markl, and Maurice van Keulen. 08421 abstracts collection–uncertainty management in information systems. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.
- [Kra97] Gerhard K Kraetzschmar. *Distributed reason maintenance for multiagent systems*, volume 1229. Springer Science & Business Media, 1997.
- [KSGM03] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.
- [Lap09] Anne Lapkin. Context-aware computing: A looming disruption. Research report, Gartner Inc., 24 August 2009.
- [LS07] Huaizhi Li and Mukesh Singhal. Trust management in distributed systems. *Computer*, (2):45–53, 2007.
- [LS12] Wolfgang Leister and Trenton Schulz. Ideas for a Trust Indicator in the Internet of Things. In *The First International Conference on Smart Systems, Devices and Technologies, SMART*, pages 31–34, 2012.
- [LV07] Ching Lin and Vijay Varadharajan. A hybrid trust model for enhancing security in distributed systems. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 35–42. IEEE, 2007.
- [LY09] Adam J Lee and Ting Yu. Towards a dynamic and composable model of trust. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 217–226. ACM, 2009.
- [Mar94] Stephen Paul Marsh. Formalising trust as a computational concept. 1994.
- [MBD12] Stephen Marsh, Anirban Basu, and Natasha Dwyer. Rendering unto caesar the things that are caesars: Complex trust models and human understanding. In *Trust Management VI*, pages 191–200. Springer, 2012.
- [MBF<sup>+</sup>12] Andreas Metzger, Adrie Beulens, Federico Facca, Fabiana Fournier, Denis Havlik, Fano Ramparany, and Zoheir Sabeur. Data and information uncertainty. *s-cube-network.eu*, 2012.

- [MS01] Yosi Mass and Onn Shehory. Distributed trust in open multi-agent systems. In *Trust in Cyber-societies*, pages 159–174. Springer, 2001.
- [NNFM12] Z Noorian, Mahdi Noorian, Michael Fleming, and Stephen Marsh. A strategic reputation-based mechanism for mobile ad hoc networks. *Advances in Artificial Intelligence*, pages 1–12, 2012.
- [Pea14] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.
- [Pła12] Bartłomiej Płaczek. Uncertainty-dependent data collection in vehicular sensor networks. In *Computer Networks*, pages 430–439. Springer, 2012.
- [PM93] Simon Parsons and EH Mamdani. On reasoning in networks with qualitative uncertainty. In *Proceedings of the Ninth international conference on Uncertainty in artificial intelligence*, pages 435–442. Morgan Kaufmann Publishers Inc., 1993.
- [PPR11] Camille Persson, Gauthier Picard, and Fano Ramparany. A multi-agent organization for the governance of machine-to-machine systems. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 421–424. IEEE Computer Society, 2011.
- [PS93] Simon Parsons and Alessandro Saffiotti. Integrating uncertainty handling formalisms in distributed artificial intelligence. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 304–309. Springer, 1993.
- [RGMSE14] Fano Ramparany, Fermin Galan Marquez, Javier Soriano, and Tarek Elsaleh. Handling smart environment devices, data and services at the semantic level with the FI-WARE core platform. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 14–20. IEEE, 2014.
- [RHJ04] Sarvapali D Ramchurn, Dong Huynh, and Nicholas R Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(01):1–25, 2004.
- [RHMV11] Sebastian Ries, Sheikh Mahbub Habib, Max Mühlhäuser, and Vijay Varadharajan. Certainlogic: A logic for modeling trust and uncertainty. In *Trust and Trustworthy Computing*, pages 254–261. Springer, 2011.

- [RPGH08] Maxim Raya, Panos Papadimitratos, Virgil D Gligor, and Jean-Pierre Hubaux. On data-centric trust establishment in ephemeral ad hoc networks. In *INFOCOM 2008. The 27th Conference on Computer Communications*. IEEE, 2008.
- [RPS<sup>+</sup>07] Fano Ramparany, Remco Poortinga, Maja Stikic, Jörg Schmalenströer, and Thorsten Prante. An open Context Information Management Infrastructure - the IST-Amigo Project. In IET: Institution of Engineering and Technology, editors, *Proceedings of the 3rd IET International Conference on Intelligent Environments (IE'07)*, pages 398–403, Germany, september 24-25 2007. University of Ulm.
- [S<sup>+</sup>76] Glenn Shafer et al. *A Mathematical Theory of Evidence*, volume 1. Princeton University Press, 1976.
- [SFP<sup>+</sup>13] Murat Sensoy, Achille Fokoue, Jeff Z. Pan, Timothy J. Norman, Yuqing Tang, Nir Oren, and Katia Sycara. Reasoning about uncertain information and conflict resolution through trust revision. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS '13*, pages 837–844. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
- [SKL09] Myra Spiliopoulou, Maurice Van Keulen, and HJ Lenz. 08421 Working Group: Imprecision, Diversity and Uncertainty: Disentangling Threads in Uncertainty Management. pages 1–3, 2009.
- [TLRJ12] WT Luke Teacy, Michael Luck, Alex Rogers, and Nicholas R Jennings. An efficient and versatile approach to trust and reputation using hierarchical bayesian modelling. *Artificial Intelligence*, 193:149–185, 2012.
- [Unc] Uncertainty Reasoning for the World Wide Web. <http://www.w3.org/2005/Incubator/urw3/XGR-urw3-20080331/Uncertainty.owl>. Accessed: 2015-11-15.
- [VRAC10] Meritxell Vinyals, Juan A Rodriguez-Aguilar, and Jesus Cernquides. A survey on sensor networks from a multiagent perspective. *The Computer Journal*, page bxq018, 2010.
- [VRJ13] Matteo Venanzi, Alex Rogers, and Nicholas R Jennings. Trust-based fusion of untrustworthy information in crowdsourcing applications. In *Proceedings of the 2013 international conference on autonomous agents and multi-agent*

- systems*, pages 829–836. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
- [WGE06] Segev Wasserkrug, Avigdor Gal, and Opher Etzion. A taxonomy and representation of sources of uncertainty in active systems. In *Next Generation Information Technologies and Systems*, pages 174–185. Springer, 2006.
- [WS07] Yonghong Wang and Munindar P Singh. Formal trust model for multiagent systems. In *IJCAI*, volume 7, pages 1551–1556, 2007.
- [XL02] Li Xiong and Ling Liu. Building trust in decentralized peer-to-peer electronic communities. In *The 5th International Conference on Electronic Commerce Research.*, 2002.
- [YC05] Yi Yang and Jacques Calmet. Ontobayes: An ontology-driven uncertainty model. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 1, pages 457–463. IEEE, 2005.
- [Zad86] Lotfi A Zadeh. A simple view of the dempster-shafer theory of evidence and its implication for the rule of combination. *AI magazine*, 7(2):85, 1986.