MONOGRAFIES DE L'INSTITUT D'INVESTIGACIÓ
EN INTEL·LIGÈNCIA ARTIFICIAL
Number 20

Institut d'Investigació
en Intel·ligència Artificial

Consell Superior
d'Investigacions Científiques

Monografies de l'Institut d'Investigació en Intel·ligència Artificial

Num. 1 J. Puyol, *MILORD II: A Language for Knowledge–Based Systems*

Num. 2 J. Levy, *The Calculus of Refinements, a Formal Specification Model Based on Inclusions*

Num. 3 Ll. Vila, *On Temporal Representation and Reasoning in Knowledge–Based Systems*

Num. 4 M. Domingo, *An Expert System Architecture for Identification in Biology*

Num. 5 E. Armengol, *A Framework for Integrating Learning and Problem Solving*

Num. 6 J. Ll. Arcos, *The Noos Representation Language*

Num. 7 J. Larrosa, *Algorithms and Heuristics for Total and Partial Constraint Satisfaction*

Num. 8 P. Noriega, *Agent Mediated Auctions: The Fishmarket Metaphor*

Num. 9 F. Manyà, *Proof Procedures for Multiple-Valued Propositional Logics*

Num. 10 W. M. Schorlemmer, *On Specifying and Reasoning with Special Relations*

Num. 11 M. López-Sánchez, *Approaches to Map Generation by means of Collaborative Autonomous Robots*

Num. 12 D. Robertson, *Pragmatics in the Synthesis of Logic Programs*

Num. 13 P. Faratin, *Automated Service Negotiation between Autonomous Computational Agents*

Num. 14 J. A. Rodríguez, *On the Design and Construction of Agent-mediated Electronis Institutions*

Num. 15 T. Alsinet, *Logic Programming with Fuzzy Unification and Imprecise Constants: Possibilistic Semantics and Automated Deduction*

Num. 16 A. Zapico, *On Axiomatic Foundations for Qualitative Decision Theory - A Possibilistic Approach*

Num. 17 A. Valls, *ClusDM: A multiple criteria decision method for heterogeneous data sets*

Num. 18 D. Busquets, *A Multiagent Approach to Qualitative Navigation in Robotics*

Num. 19 M. Esteva, *Electronic Institutions: from specification to development*

Num. 20 J. Sabater, *Trust and reputation for agent societies*

Trust and reputation for agent societies

Jordi Sabater i Mir

Foreword by Carles Sierra i Garcia

2003 Consell Superior d'Investigacions Científiques
Institut d'Investigació en Intel·ligència Artificial
Bellaterra, Catalonia, Spain.

Series Editor
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Foreword by
Carles Sierra i Garcia
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Volume Author
Jordi Sabater i Mir
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Institut d'Investigació
en Intel·ligència Artificial

Consell Superior
d'Investigacions Científiques

NIPO: 403-03-085-8
ISBN: 84-00-08157-9
Dip. Legal: B.43079-2003

Ordering Information: Text orders should be addressed to the Library of the IIIA, Institut d'Investigació en Intel·ligència Artificial, Campus de la Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, Spain.

Als meus pares.

Nothing in life is to be feared,
it is only to be understood.

(Marie Curie)

Contents

List of Figures

Foreword

This book introduces one of the currently most sophisticated computational models of trust. It goes beyond the current state-of-the-art by bridging numerical methods to model reputation and the social dimension of interacting agents. Jordi Sabater's model is very innovative by mixing techniques from Sociology and from Utility theory. The current evolution of Internet and electronic commerce requires of new methods to guarantee the outcome of automated transactions. Jordi's work explores specifically the electronic commerce dimension of agent interaction. He proposes a benchmark to compare different computation models and uses it to experimentally show the advantages of using REGRET, his model. The reader will also find a complete and solid state-of-the-art description of the current computational models of trust that permits to perceive REGRET on an outstanding position among them.

Jordi Sabater managed to develop his Ph. D. concurrently with the heavy task of working in the context of a European research project. This has a lot of merit. Not only that, his many personal qualities permitted him to devote time to splendid social activities within the IIIA, specially providing an excellent musical level to our Christmas Concerts. I'm sure that his personal interest for music and culture were driving forces to use sociological concepts in his approach. You can feel how electronic commerce is humanised along the book. It is a very interesting piece of work that I'm sure the reader will enjoy.

Bellaterra, July 1998

Carles Sierra i Garcia
IIIA-CSIC

Acknowledgements

Una tesi doctoral és un camí llarg i difícil, ple d'alegries però també de dificultats i frustracions. Per sort aquest camí no es fa mai sol.

Vull començar aquests agraïments per la persona que sens dubte a contribuit de manera més directa a què aquesta història tingués un final feliç. Aquesta persona no és altre que en Carles Sierra. A més de ser el millor director de tesi que hom podria desitjar és un bon amic amb qui he compartit més d'una cervesa.

També vull donar molt especialment les gràcies:

- a en Jaume Agustí (projecte SLIE) i a en Lluís Godo (projecte SMASH) per haver confiat en mi. Sense aquesta confiança hauria estat impossible portar a bon terme aquesta tesi.
- to Professor Nick Jennings and Dr. Simon Parsons for their invaluable contributions to chapter 5. It has been a privilege to work with you.
- als meus col.legues de l'IIIA pels seus consells, les xerrades i els bons moments que m'han fet passar.
- to my collegues in the university of Edinburgh, Dave Robertson and Chris Walton and to my friend Wamberto Vasconcelos (now in the university of Aberdeen). The SLIE project has been the best school to learn how to become a real scientist.
- a tots els meus amics. Per les bones estones que m'heu fet passar.

Finalment vull donar les gràcies als meus pares. Si a algú he d'agrair l'haver arribat fins aquí és a vosaltres que, en els moments més difícils, sempre heu estat al meu costat donant-me suport.

Aquest treball ha estat finançat pel CSIC i l'Hospital de Mataró a través del projecte SMASH (TIC96-1038-C04001), i per la Comunitat Europea a través del projecte SLIE (IST-1999-10948). El meu sincer agraïment a aquestes institucions.

Abstract

The area of trust and reputation mechanisms for virtual societies is a recent discipline oriented to increase the reliability and performance of electronic communities by introducing in such communities these well known human social control mechanisms.

This thesis has two different parts. In the first part we present ReGreT, a sophisticated trust and reputation system oriented to complex societies where the social component of the agents behaviour has a special relevance. The system uses the knowledge about the social structure of the society as a method to overcome the lack of direct experiences and to evaluate the credibility of witnesses. By combining direct experiences, third party information and social knowledge, the system can improve the computation of trust and reputation values. It also provides a degree of reliability for these values and can adapt to situations of partial information improving gradually its accuracy when new information becomes available.

A trust and reputation system like ReGreT only has sense if it is used by a deliberative agent. The second part of the thesis is devoted to the use of *multi-context* systems as a means of specifying and implementing this kind of agents. We propose several extensions to the *multi-context* approach that facilitate the specification and implementation of complex deliberative agents. As the main example of using *multi-context* systems to specify deliberative agents we present the specification of an agent's component implementing the ReGreT system.

Chapter 1

Introduction

1.1 Motivation

The sociologist Niklas Luhmann said "A complete absence of trust would prevent [one] even getting up in the morning" [Luhmann, 1979]. *Trust* is necessary in our everyday life. It is part of the "glue" that holds our society together. Without trust, governments could not rule and people cannot work cooperatively together. Trust helps to reduce the complexity of decisions that have to be taken in the presence of many risks.

Similarly, *reputation* is a universal concept that has been present in human societies for a long time. From ancient Greeks to modern days, from Vietnamese to Bedouins, the concept of reputation plays a very important role in human social organization. Reputation is one of the most relevant elements that we use to build trust in others.

Until recently, both concepts were applicable only to human societies and therefore they were a study field for sociologists, philosophers and psychologists. The irruption of Internet and the emergence of virtual (not necessarily human) societies add a new dimension to these old but very important concepts.

The scientific research in the area of trust and reputation mechanisms for virtual societies is a recent discipline oriented to increase the reliability and performance of electronic communities by introducing in such communities these well known human social control mechanisms. Computer science has moved from the paradigm of an isolated machine to the paradigm of a network of systems and of distributed computing. Likewise, artificial intelligence is quickly moving from the paradigm of an isolated and non-situated intelligence to the paradigm of situated, social and collective intelligence. The new paradigm of the so called intelligent or adaptive agents and Multi-Agent Systems (MAS) together with the spectacular emergence of the information society technologies (specially reflected by the popularization of electronic commerce) are responsible for the increasing interest on trust and reputation mechanisms applied to electronic societies.

An agent is a computer system capable of flexible autonomous action in a dynamic, unpredictable and open environment endowed with the capacity to interact with other systems (artificial or natural). Agents are often deployed in environments in which they

interact, compete, and maybe cooperate, with other agents that have possibly conflicting aims. Such environments are known as multi-agent systems and are called to become a key element of the information society. In this context, trust and reputation play a similar role that in human societies.

Up to now, the design of such systems has been approached using traditional software development methods. However, the special characteristics of these systems suggest the necessity of more specific techniques adapted to its peculiarities.

1.2 Overview and main contributions

The work of this thesis contributes to the state of the art in two areas.

First, in the area of computational trust and reputation models, we present ReGreT. ReGreT is a modular trust and reputation system oriented to complex e-commerce environments where social relations play an important role. ReGreT follows a mathematical approach to the problem, with social ingredients that improve the calculation of trust and reputation values. There is a lot of work done in computational trust and reputation models, however little attention has been given to the social aspect of both concepts. With our model we want to strengthen a line of research that we think is under-explored and very promising.

The main characteristics of the ReGreT system can be summarized as follows:

- It takes into account direct experiences, information from third party agents and social structures to calculate trust, reputation and credibility values.
- It has a trust model based on direct experiences and reputation.
- It incorporates an advanced reputation model that works with transmitted and social knowledge.
- It has a credibility module to evaluate the truthfulness of information received from third party agents.
- It uses social network analysis to improve the knowledge about the surrounding society (specially when no direct experiences are available).
- It provides a degree of reliability for the trust, reputation and credibility values that helps the agent to decide if it is sensible or not to use them in the agent's decision making process.
- It can adapt to situations of partial information and improve gradually its accuracy when new information becomes available.
- It can manage at the same time different trust and reputation values associated to different behavioural aspects. Also it can combine reputation and trust values linked to simple aspects in order to calculate values associated to more complex attributes.

The study of trust and reputation has many applications in Information and Communication technology. Trust and reputation systems have been recognized as key factors for successful electronic commerce adoption. These systems are used by intelligent software agents both as a mechanism of search for trustworthy exchange partners and as an incentive in decision-making about whether or not to honour contracts. Reputation is used in electronic markets as a trust-enforcing, deterrent, and incentive mechanism to avoid cheaters and frauds. Another area of application in agent technology is teamwork and cooperation.

Besides technological issues, if we want people to massively enter the information society era, we have to improve the real and perceived security of electronic interactions. For instance, it is well known that lack of trust is one of the main reasons for consumers, as well as companies, not engaging in electronic commerce. This lack of confidence is even worse when users have to rely on autonomous agents that act on their behalf. With no doubt, low level security measures are important and necessary. However, building user confidence in e-Commerce (and electronic interactions in general) is more than secure communication via electronic networks as can be obtained with, for example public key cryptography techniques. Here is where trust and reputation mechanisms come to scene. As in human societies, electronic communities have started to use trust and reputation as a social control mechanism that complements more expeditious approaches (based on social constructs and their corresponding disciplinary actions to punish deception and fraud). Further study on trust and reputation mechanisms will directly contribute increasing both the reliability and performance of electronic communities and the confidence that humans deserve on information society technologies.

The second contribution is in the area of agent design. As we have pointed out in the motivations section, traditional software development methods do not fully cover the requirements for the design and implementation of autonomous agents. Several approaches have been proposed to overcome this problem. However these proposals usually leave a gap between specification and implementation, enforcing a particular view of architecture upon the specification or do not make explicit structures for modelling the components of an architecture or the relationships between them.

Following the steps of Parsons, Sierra and Jennings [Parsons et al., 1998] we propose the use of multi-context systems for the design and implementation of autonomous agents.

Multi-context systems provide an overarching framework that allows distinct theoretical components to be defined and interrelated. Such systems consists of a set of contexts, each of which can informally be considered a set of formulae written in a (possibly) different logic, and a set of bridge rules for transferring information between contexts. From a software engineering perspective, multi-context systems support modular decomposition and encapsulation but what is more important for our purposes, from a logical modelling perspective they provide an efficient means of specifying and executing complex logics. We extend the multi-context theory with several elements that make the specification of dynamic components, as those needed to build autonomous agents, easier.

As a nexus between both contributions, we show how it is possible to specify the

ReGreT system using the multi-context approach.

1.3 Publications

The work presented in this thesis has generated the following set of publications:

- J. Sabater, C. Sierra, S. Parsons, N. Jennings (2002) Engineering executable agents using multi-context systems, Journal of Logic and Computation. Vol 12, n 3, pp. 413-442.

- J. Sabater, C. Sierra (2002) Reputation and social network analysis in multi-agent systems, Proc. "First International Conference on Autonomous Agents and Multiagent systems (AAMAS-02)", Bologna, Italy, (July 15-19), pp. 475-482.

- J.Sabater, C. Sierra (2002) Social aspects or ReGreT, a reputation model based on social relations, Proc. "5è Congrés Català d'Intel.ligència Artificial (CCIA-02)", Castelló de la Plana, Spain, (October 24-25), pp. 336-343

- J. Sabater, C. Sierra (2002) Social ReGreT, a reputation model based on social relations, SIGecom Exchanges. ACM, Vol 3.1, pp 44-56. (http://www.acm.org/sigecom/exchanges/volume_3_(02)/3.1-Sabater.pdf)

- J. Sabater, C. Sierra (2001) ReGreT: A reputation model for gregarious societies (v.1), Proc. "Fourth Workshop on Deception Fraud and Trust in Agent Societies", Montreal, Canada, (May 28, June 9), pp. 61-70.

  (Also published at Proc. "4t Congrés Català d'Intel.ligència Artificial", Barcelona, Spain, (October 24-25), pp. 214-222)

- J. Sabater, C. Sierra (2001) ReGreT: A reputation model for gregarious societies (v.2), Proc. "Fifth International Conference on Autonomous Agents", Montreal, Canada, (May 28, June 9), pp. 194-195

- J. Sabater, C. Sierra, S. Parsons, N. R. Jennings (1999) Using multi-context systems to engineer executable agents Proc. "6th Int.Workshop on Agent Theories Architectures and Languages (ATAL-99)", Orlando, Florida, USA, (July 15-17), pp. 131-148.

  Revised version in: Intelligent Agents VI (eds N. R. Jennings and L. Lesperance) LNAI 1757 pp. 277-294.

  (Also published at UKMAS 99 (2nd workshop of the UK special interest group on multi-agent systems, Bristol, UK, (December 6-7)))

  (Reduced version published -in Catalan- at "2n Congrés Català d'Intel.ligència Artificial", Girona, Spain, (October 25-27), pp. 185-191.)

1.4 Structure of the thesis

The thesis is organized in eight chapters and two appendices that are distributed in three blocks: related work (chapter 2), the ReGreT system (chapters 3, 4, 7 and appendix A) and the use of multi-context systems to engineer autonomous agents (chapters 5, 6 and appendix B).

Chapter 2: we analyse a representative set of computational trust and reputation models as well as some frameworks/test-beds currently used to evaluate these models. We also propose a classification of the models according to a set of relevant aspects associated to trust and reputation.

Chapter 3: this chapter introduces the *SuppWorld* framework, a flexible framework specially designed to test trust and reputation models in a complex environment where social relations play an important role.

Chapter 4: in this chapter *ReGreT* is presented, a trust and reputation system that gives special relevance to the social relationships among individuals in virtual societies.

Chapter 5: is devoted to the use of multi-context systems as a means of specifying and implementing agent architectures. We propose some extensions to the multi-context base theory and, by means of two didactic examples, we show how it works in practice.

Chapter 6: this chapter puts together the two main threads of this thesis: the trust and reputation models and the multi-context approach for the design of autonomous agents. We take the *ReGreT* system described in chapter 4 and specify it by using the multi-context approach presented in chapter 5.

Chapter 7: an initial set of experimental results for the ReGreT system are presented in this chapter. Using the framework described in chapter 3 we deploy a set of scenarios that are used to test the ReGreT system capabilities.

Chapter 8: summarizes the conclusions of this work and shows future directions.

Appendix A: this appendix presents the unit theories of the *ReGreT* system specification made in chapter 6.

Appendix B: this appendix details the *SuppWorld* framework configuration files used in the experiments of chapter 7.

Chapter 2

Related work on computational trust and reputation models

2.1 Introduction

It is out of discussion the importance of trust and reputation in human societies. Therefore, it is not a surprise that several disciplines, each one from a different perspective, have studied and used both concepts. Psychology [Bromley, 1993, Karlins and I.Abelson, 1970], sociology [Buskens, 1998], philosophy [Plato, 1955, Hume, 1975] and economy [Marimon et al., 2000, Celentani et al., 1966] are a good representation of disciplines that have dedicated efforts to the study of trust and reputation. In this overview, however, we will focus our attention on another discipline where the study of trust and reputation has acquired a great relevance in the last few years. We are talking about computer science and specifically about the area of distributed AI. Two elements have contributed to substantially increase the interest on trust and reputation in this area: the multi-agent system paradigm and the spectacular evolution of e-commerce.

This review tries to offer a panoramic view on current computational trust and reputation models. We propose a set of relevant aspects associated to both concepts that are then used to establish a classification of these models. We present a representative selection of models that, although far from being exhaustive, gives a rather complete idea of the current state of the art.

The second part of this review is devoted to present some of the test-beds used to evaluate and compare trust and reputation models. Currently, there is no test-bed (or set of them) accepted by all the members of the community. That would make the comparison between models easier and more neutral. Unfortunately, virtually every author uses a different test-bed to show the properties of his/her model and to establish comparisons with others models. In section 2.4 we analyze a representative group of these test-beds.

There are not many works that give a general view of trust and reputation from the point of view of computer science. Dellarocas in his article "The digital-

ization of Word-Of-Mouth: Promise and Challenges of Online Reputation Mechanisms" [Dellarocas, 2002] presents an overview of online reputation mechanisms that are currently used in commercial web sites. In the area of trust, Grandison et al. in their work "A survey of trust in Internet application" [Grandison and Sloman, 2000] examine the various definitions of trust in the literature and provide a working definition of trust for Internet applications. There are also some proposals to establish a typology for reputation [Mui et al., 2002] and trust [McKnight and Chervany, 2002].

2.2 Classification dimensions

Trust and reputation can be analyzed from different perspectives and can be used in a wide range of situations. This makes the classification of trust and reputation models a difficult task. In this section we propose a set of aspects with which we classify the current computational trust and reputation models in a clear landscape.

2.2.1 Paradigm type

Two different paradigms are currently used in computational trust and reputation models: a cognitive approach and a numerical approach. As pointed out in [Esfandiari and Chandrasekharan, 2001], in models based on a cognitive view, trust is made up of underlying beliefs, and trust is a function of the degree of these beliefs. On the opposite side we have the numerical based models. These models do not rely on beliefs nor intentions to model trust and reputation. Trust and reputation are not the result of a mental process of the agent in a cognitive sense but the result of a more pragmatic game with utility functions, probabilities and the evaluation of past interactions. The cognitive approach tries to reproduce the human reasoning mechanisms behind trust and reputation and the process of building trust and reputation on others is as important as its outcome.

2.2.2 Information sources

It is possible to classify trust and reputation models considering the information sources that they take into account to calculate trust and reputation values. Direct experiences and witness information are the "traditional" information sources used by computational trust and reputation models. In addition to that, recently a few models have started to use information associated to sociological aspects of agent's behaviour.

The kind of information available to an agent depends on its sensorial capabilities. Of course, the scenario must be in tune with these capabilities. The use of several information sources increases the reliability of the calculated trust and reputation values but also increases the complexity of the model. Moreover, scenarios that allow agents to obtain diverse information demand smarter (and, therefore, more complex) agents.

Direct experiences

This is, without doubt, the most relevant and reliable information source for a trust/reputation model. There are two types of direct experiences that an agent can include as part of its knowledge. The first, and used by all the trust and reputation models analyzed in this overview, is the experience based on the direct interaction with the partner. The second is the experience based on the observed interaction of other members of the community. This second type is not so common and restricted to scenarios that are prepared to allow it. Usually, in those models that consider the observation of other partners activity, a certain level of noise in the so obtained information is assumed.

Witness information

Witness information (also called word-of-mouth or indirect information) is the information that comes from other members of the community. That information can be based on their own direct experiences or it can be information that they gathered from others. If direct experience is the most reliable source of information for a trust/reputation model, witness information is usually the most abundant. However, it is far more complex for trust and reputation models to use it. The reason is the uncertainty that surrounds this kind of information. It is not strange that witnesses manipulate or hide pieces of information to their own benefit.

Sociological information

This information is only available in scenarios where there is a rich interaction between agents. The base for this knowledge are the social relations between agents and the role that these agents are playing in the society. The social relations established between agents in a multi-agent system are usually a simplified reflection of the more complex relations established between their human counterparts. Currently, only a few trust and reputation models use this knowledge applied to agent communities to calculate or improve the calculation of trust and reputation values. These models use techniques like *social network analysis*. Social network analysis is the study of social relationships between individuals in a society that emerged as a set of methods for the analysis of social structures, methods that specifically allow an investigation of the relational aspects of these structures. The use of these methods, therefore, depends on the availability of relational data [Scott, 2000].

Although currently the number of models that take into account this kind of information is reduced, we guess that the increase of complexity in multi-agent systems will make it more and more important in the near future.

Prejudice

The use of prejudice to calculate trust and reputation values is another mechanism not very common but present in current trust and reputation models. As most people today use the word, "prejudice" refers to a negative or hostile attitude toward another social group, usually racially defined. However, the negative connotations that has prejudice in human societies has to be revised when applied to agent communities. Differently

from the signifiers used in human societies that range from skin color to sex, the set of signifiers used in computational trust and reputation models are usually out of ethical discussion.

2.2.3 Visibility types

Trust and reputation of an individual can either be seen as a global property shared by all the observers or as a subjective property assessed particularly by each individual.

In the first case, the trust/reputation value is calculated from the opinions of the individuals that in the past interacted with the individual being evaluated. This value is publicly available to all members of the community and updated each time a member issues a new evaluation of an individual. In the second case, each individual assigns a personalized trust/reputation value to each member of the community according to more personal elements like direct experiences, information gathered from witnesses, known relations between members of the community and so on. In the latter case, we cannot talk about the trust/reputation of an individual x, we have to talk about the trust/reputation of an individual x from the point of view of an individual y.

The position of taking trust and reputation as a global property is common in online reputation mechanisms (see section 2.3.2). These systems are intended for scenarios with thousands or even millions of users. As pointed out by Dellarocas [Dellarocas, 2002], the size of these scenarios makes repeated interaction between the same set of players unlikely and, therefore, reduces the incentives for players to cooperate on the basis of hoping to develop a profitable relationship.

Take the example of an electronic auction house like those accessible nowadays through Internet. One day, the user wants to buy a book and the next day s/he wants to buy a computer. The intersection between users selling books and users selling computers is probably empty so the few personal experiences accumulated buying books are not useful in the computers market. Computer sellers are unknown for the user so s/he has to rely on the information that people who bought computers in the past has left in the form of a reputation value. The robustness of these systems relies on the number of opinions available for a given partner. A great number of opinions minimize the risk of single individual biased perceptions.

In models that consider trust and reputation as a global property, the main problem is the lack of personalization of that value. Something that is bad for me could be acceptable for others and the other way around. Although this approach can be acceptable in simple scenarios where it is possible to assign a common "way of thinking" to all members of the community, it is not useful when agents have to deal with more complex and subjective affairs.

The antithesis of these models are the models that consider trust and reputation as a subjective property. Each agent uses its personal experiences and what the other agents have said to it personally, among other things, to build the trust and reputation of each member of the community. These models are indicated for medium and small size environments where agents meet frequently and therefore it is possible to establish strong links among them.

2.2.4 Model's granularity

Is trust/reputation context dependent? If we trust a doctor when she is recommending a medicine it doesn't mean we have to trust her when she is suggesting a bottle of wine. The reputation as a good sportsman does not help if we are looking for a competent scientist. It seems clear that the answer is yes: trust and reputation are context dependent properties. However, adding to computational trust and reputation models the capability to deal with several contexts has a cost in terms of complexity and adds some side effects that are not always necessary or desirable.

A single-context trust/reputation model is designed to associate a single trust/reputation value per partner without taking into account the context. A multi-context model has the mechanisms to deal with several contexts at a time maintaining different trust/reputation values associated to these contexts for a single partner.

One could argue that it is always possible to transform a single-context model into a multi-context one just having different instances of the single-context model, one for each considered context. However, if there is something in trust and reputation environments that is usually scarce, that is the information used to calculate trust and reputation values. So what really gives to a model the category of being a multi-context model is the capability of making a smart use of each piece of information to calculate different trust or reputation values associated to different activities. Identifying the right context for a piece of information or using the same information in several contexts when it is possible are two examples of the capabilities that define a real multi-context model.

Is this always necessary? Certainly not. Nowadays, there are very few computational trust and reputation models that care about the multi-context nature of trust and reputation and even fewer that propose some kind of solution. This is because current models are focused on specific scenarios with very delimited tasks to be performed by the agents. In other words, it is possible to summarize all the agent activities in a single context without losing too much versatility. However, and similarly to what we have mentioned before about the use of sociological information, as the complexity of tasks to be performed by agents will increase in the near future, we may also expect also an increase of the importance devoted to this aspect in trust modelling.

2.2.5 Agent behaviour assumptions

The capacity to deal with agents showing different degrees of cheating behaviour is the aspect considered here to establish a classification. We use three levels to categorize trust and reputation models from this point of view according to what we have observed in the analyzed trust and reputation models:

- Level 0. Cheating behaviour is not considered. They rely on a large number of agents who offer honest ratings to counteract the potential effect of the ratings provided by malicious agents.

- Level 1. The model assumes that agents can hide or bias the information but they never lie.

- Level 2. The model has specific mechanisms to deal with liars.

2.2.6 Type of exchanged information

The classification dimension here is the type of information expected from witnesses. We can establish two big groups. Those models that assume boolean information and those models that deal with continuous measures. Although it seems a trivial assumption, as we will see, choosing one approach or the other has a great influence in the design of the model. Usually, models that rely on probabilistic methods work with boolean information while those models based on aggregation mechanisms use continuous measures.

2.3 Computational trust and reputation models

A plethora of computational trust and reputation models have appeared in the last few years, each one with its own characteristics and using different technical solutions. In this section we go through a selection of these models, wide enough to provide a panoramic view of the area.

2.3.1 S. Marsh

The trust model proposed by Marsh [Marsh, 1994] is one of the earliest. The model only takes into account direct interaction. It differentiates three types of trust:

- Basic trust. Models the general trusting *disposition* independently of who is the agent that is in front. It is calculated from all the experiences accumulated by the agent. Good experiences lead to a greater disposition to trust, and vice versa. The author uses the notation T_x^t to represent the trust disposition of agent x at time t.

- General trust. This is the trust that one agent has on another without taking into account any specific situation. It simply represents general trust in the other agent. It is noted as $T_x(y)^t$ representing the general trust that agent x has on agent y at time t.

- Situational trust. This is the amount of trust that one agent has in another taking into account a specific situation. The basic formula used to calculate this type of trust is:

$$T_x(y,\alpha)^t = U_x(\alpha)^t \times I_x(\alpha)^t \times \widehat{T_x(y)^t}$$

  where x is the evaluator, y the target agent and α the situation. $U_x(\alpha)^t$ represents the utility x gains from situation α, $I_x(\alpha)^t$ is the importance of the situation α for agent x and $\widehat{T_x(y)^t}$ is the estimate of general trust after taking into account all possible relevant data with respect to $T_x(y,\alpha)$ in the past; i.e., if t is the current time, x will aggregate all situations $T_x(y,\sigma)^T$, with $\theta < T < t$ and σ similar or identical to the present situation α. θ and t define the temporal window that the agent is considering. Only the experiences within that window will be taken into account for the aggregation.

In order to define $\widehat{T_x(y)}$ the author proposes three statistical methods: the mean, the maximum and the minimum. Each method is identified with a different type of agent: the optimistic (that takes the maximum trust value from the range of experiences it has had), the pessimistic (that uses the minimum trust value) and the realistic (that calculates the value as a mean using the formula $\widehat{T_x(y)} = \frac{1}{|A|} \sum_{\alpha \in A} T_x(y, \alpha)$, where A is the set of situations similar to the present situation α available in the temporal window).

These trust values are used to help an agent decide if it is worth it or not to co-operate with another agent. Besides trust, the decision mechanism takes into account the importance of the action to be performed, the risk associated to the situation and the perceived competence of the target agent. To calculate the risk and the perceived competence, different types of trust (basic, general and situational) are used.

Finally, the model also introduces the notion of "reciprocation" as a modifier of the trust values. The idea behind reciprocation is that if an agent x had helped an agent y in the past and y responded that time by defecting, the trust x has on y will be reduced (and the other way around).

2.3.2 Online reputation models

eBay [eBay, 2002], Amazon Auctions [Amazon, 2002] and OnSale Exchange [OnSale, 2002] are good examples of online marketplaces that use reputation mechanisms. eBay [eBay, 2002] is one of the world's largest online marketplace with a community of over 50 million registered users. Most items on eBay are sold trough English auctions and the reputation mechanism used is based on the ratings that users perform after the completion of a transaction. The user can give three possible values: *positive*(1), *negative*(-1) or *neutral*(0). The reputation value is computed as the sum of those ratings over the last six months. Similarly, Amazon Auctions [Amazon, 2002] and OnSale Exchange [OnSale, 2002] use also a mean (in this case of all ratings) to assign a reputation value.

All these models consider reputation as a global property and use a single value that is not dependent on the context. The information source used to build the reputation value is the information that comes from other agents that previously interacted with the target agent (witness information). They do not provide explicit mechanisms to deal with users that provide false information. A great number of opinions that "dilute" false or biased information is the only way to increase the reliability of the reputation value.

2.3.3 Sporas and Histos

Sporas

Sporas [Zacharia, 1999] is an evolved version of the online reputation models presented in 2.3.2. In this model, only the most recent rating between two users is considered. Another important characteristic is that users with very high reputation values experience much smaller rating changes after each update than users with a low reputation. Using

a similar approach to the Glicko [Glickman, 1999] system —a computational method used to evaluate the player's relative strengths in pairwise games—, Sporas incorporates a measure of the reliability of the users' reputation based on the standard deviation of reputation values.

This model has the same general characteristics as the previously commented online reputation mechanisms 2.3.2. However, it is more robust to changes in the behaviour of a user and the reliability measure improves the usability of the reputation value.

Histos

Histos [Zacharia, 1999] was designed as a response to the lack of personalization that Sporas reputation values have. The model can deal with direct information (although in a very simple way) and witness information. Contrary to Sporas, the reputation value is a subjective property assigned particularly by each individual.

The treatment of direct interaction in this reputation model is limited to the use of the most recent experience with the agent that is being evaluated. The strength of the model relies on its use of witness information.

Pairwise ratings are represented as a directed graph where nodes represent agents and edges carry information on the most recent reputation rating given by one agent to another. The root node represents the agent owner of the graph. This structure is similar to the *TrustNet* used by Schillo et al. [Schillo et al., 2000]. The reputation of an agent at level X of the graph (with $X > 0$) is calculated recursively as a weighted mean of the rating values that agents in level X-1 gave to that agent. The weights are the reputations of the agents that rate the target agent. As we have seen, the agents who have been rated directly by the agent owner of the graph have a reputation value equal to the rating value. This is the base case of the recursion. The model also limits the length and number of paths that are taken into account for the calculation. The reputation value does not depend on the context and no special mechanisms are provided to deal with cheaters.

A drawback of this model is the use of the reputation value assigned to a witness also as a measure of its reliability. If an agent is a good seller, it does not mean that it has to be also a reliable witness.

2.3.4 Schillo et al.

The trust model proposed by Schillo et al. [Schillo et al., 2000] is intended for scenarios where the result of an interaction between two agents (from the point of view of trust) is a boolean impression: good or bad; there are no degrees of satisfaction. More concretely, to make the experiments they propose a Prisoner's dilemma set of games with a partner selection phase (see 2.4.1 for more details). Each agent receives the results of the game it has played plus the information about the games played by a subset of all players (its neighbours). The result of an interaction in this scenario is an impression on the honesty of the partner (if she did what she claimed in the partner selection phase) and which was the behaviour she had according to the normal prisoner's dilemma actions (cooperation or defection). The model is based on probability theory. The formula to calculate the trust that an agent Q deserves to an agent A (that is, the probability that

the agent A be honest in the next interaction) is $T(A, Q) = \frac{e}{n}$ where n is the number of observed situations and e the number of times that the target agent was honest.

Complementing the information that results from direct interaction/observation, an agent can interview other agents that it has met before. Each agent uses a different *TrustNet* data structure. A *TrustNet* is a directed graph where nodes represent witnesses and edges carry information on the observations that the parent node agent told the owner of the net (the root node of the *TrustNet*) about the child node agents.

In this model, testimonial evidence from interviews may be brittle, as witnesses may have different motives and may try to deceive agents about their true observation. Thus, every agent is confronted with noise in the information and also with the possibility that the source of information itself is biasing the data.

The answer of witnesses to a query is the set of observed experiences (and not a summary of them). Given that, the authors assume that it is not worth it for witnesses to give *false* information. A witness will not say that a target agent has played dishonest in game x if this was not the case because the inquirer could have observed the same game and, therefore, notice that the witness is lying. Witnesses do not want to be uncovered by obviously betraying. Therefore, the model assumes that witnesses never lie but that can hide (positive) information in order to make other agents appear less trustworthy. Assuming that negative information will be always reported by witnesses, the problem is reduced to know to what extend those witnesses have biased the reported data (hiding positive observations).

To do that, betraying (hiding information) is modelled as a stochastic process where an agent decides to inform about a positive fact of another agent with probability p and hide that information with probability $(1 - p)$. The application of this process can be seen as a Bernoulli-experiment and the repetition of the experiment as a Bernoulli-chain. Probability theory is then used to estimate the hidden amount of positive information. This process can be applied recursively from the target agent through all its ancestors up to the root node of the *TrustNet*.

With all this process, the agent is building for each piece of information an approximation of what the witnesses would have said if they had been completely honest about their information.

As the information from the witnesses comprises the list of observations it can be collated to eliminate the "correlated evidence" problem [Pearl, 1988]. This, however, cannot be done for the hidden information. The proposed solution in this case is based on the assumption that the relation of overlapping of the data in reported an non reported (hidden) information is constant.

No information is given about how to combine direct experiences with information coming from witnesses.

The trust value is a subjective property assigned particularly by each individual and it does not depend on the context.

2.3.5 Abdul-Rahman and Hailes

This trust model [Abdul-Rahman and Hailes, 2000] uses four degrees of belief to typify agent trustworthiness: *vt* (very trustworthy), *t* (trustworthy), *u* (untrustworthy) and *vu* (very untrustworthy). For each partner and context, the agent maintains a tuple with the

number of past experiences in each category. Then, from the point of view of direct interaction, the trust on a partner in a given context is equal to the degree that corresponds to the maximum value in the tuple. For instance if the associated tuple of a partner in a given context is $(0, 0, 4, 3)$ the trust assigned to that partner will be *t* (trustworthy) that corresponds to the third position in the tuple. If there is more than one position in the tuple with the maximum value, the model gives an *uncertainty* trust degree according to a table of pattern situations that cover this cases. There are three possible uncertainty values (and the corresponding patterns) to cover the situations where there are mostly good and some bad, mostly bad and some good and equal amount of good and bad experiences.

This is the only model analyzed where before combining the information that comes from witnesses, the information is adjusted according to previous information coming from that witness and the consequent outcomes that validate that information. For example, suppose a informs to x that b is *vt* and x's evaluation of its experience with b is merely *t*. Next time that a gives information to x, x will adjust the information accordingly before taking it into account.

The problem of this approach is that it is not possible to differentiate those agents that are lying from those agents that are telling the truth but "think" different. Although there are scenarios where this is not important (like the scenario suggested by the authors where agents recommend goods to other agents) it can be a limitation in others.

In order to combine information, the model gives more relevance to the information coming from those agents with a more similar point of view. That is, it gives more importance to the information that needs to be adjusted very little or, even better, do not need to be adjusted at all because it comes from agents that have a similar perspective in a given context.

Contrarily to other trust models where witness information is merged with direct information to obtain the trust on the specific subject, this model is intended to evaluate only the trust on the information given by witnesses. Direct experiences are used to compare the point of view of these witnesses with the direct perception of the agent and then be able to adjust the information coming from them accordingly.

2.3.6 Esfandiary and Chandrasekharan

In the trust model proposed by Esfandiari and Chandrasekharan [Esfandiari and Chandrasekharan, 2001], two one-on-one trust acquisition mechanisms are proposed. The first is based on observation. They propose the use of Bayesian networks and to perform the trust acquisition by Bayesian learning. In the simplest case of a known structure and a fully observable Bayesian network, the learning task is reduced to statistical considerations.

The second trust acquisition mechanism is based on interaction. The approach is the same used in [Lashkari et al., 1994]. There are two main protocols of interaction, the *exploratory protocol* where the agent asks the others about known things to evaluate their degree of trust and the *query protocol* where the agent asks for advice from trusted agents. A simple way to calculate the interaction-based trust during the exploratory stage is using the formula $T_{inter}(A, B) = \frac{\text{number\_of\_correct\_replies}}{\text{total\_number\_of\_replies}}$.

To deal with witness information, each agent builds a directed labeled graph where nodes represent agents and where an (a,b) edge represents the trust value that a has on b. Edges are absent if the trust value is unknown. In such a graph, there is the possibility of having cycles that artificially decrease the trust value and different paths that give contradictory values. To solve this problem, instead of using a single value for trust the model uses a trust interval determined by the minimum and maximum value of all paths without cycles that connect two agents.

The authors claim that the calculation of this trust interval is equivalent to the problem of routing in a communication network and, therefore, known distributed algorithms used to solve that problem can be successfully applied to this situation.

To allow a multi-context notion of trust (see section 2.2.4) the authors propose the use of colored edges, with a color per task or type of trust. Trust would only propagate through edges of the same color.

Finally, the authors propose a trust acquisition mechanism using institutions, what they call *institutionalized trust*. This is similar to the concept of *system reputation* in the ReGreT [Sabater and Sierra, 2002] model. The idea is to exploit the structure in the environment to arrive at trust values.

No information is given about how to combine the different trust acquisition mechanisms.

2.3.7 Yu and Singh

In the model proposed by Yu and Singh [Yu and Singh, 2001, Yu and Singh, 2002b, Yu and Singh, 2002a], the information stored by an agent about direct interactions is a set of values that reflect the quality of these interactions (what they call quality of service -*QoS*-). Only the most recent experiences with each concrete partner are considered for the calculations. Each agent defines an upper and lower threshold that define the frontier between what are considered *QoS*s ascribed to trustworthy agents, *QoS*s with no clear classification and *QoS*s ascribed to non trustworthy agents. Then, using the historic information together with Dempster-Shafer theory of evidence, an agent can calculate the probability that its partner gives a service ascribed to each one of these groups. If the difference between the probability that the service belongs to the first and latest group is greater than a threshold for trustworthiness, the agent being evaluated is considered a trusty agent.

There are two kinds of information that a witness can provide when it is queried about a target agent. If the target agent is one of its acquaintances it will return the information about it. If not, it will return referrals to the target agent that can be queried to obtain the information. These referrals, when queried, can provide the desired information or provide again new referrals. If the referral that finally gives the information is not far away to a depth limit in the chain, its information will be taken into account. The set of referral chains generated due to a query is a *TrustNet* similar to that used by Schillo et al. [Schillo et al., 2000] and in the Histos [Zacharia, 1999] model.

As we have said this model uses Dempster-Shafer theory of evidence as the underlying computational framework. In this case, to aggregate the information from different witnesses they use Dempster's rule of combination.

This model does not combine direct information with witness information (the two sources of information that takes into account). If direct information is available, that's the only source that is considered to determine the trust of the target agent. Only when direct information is not available the model appeals to witness information.

2.3.8 Sen and Sajja

In Sen and Sajja's [Sen and Sajja, 2002] reputation model, both types of direct experiences are considered: direct interaction and observed interaction. In the scenario where this model is used, observations are noisy, i.e., the observations differ somewhat from the actual performance. Only direct interaction gives an exact perception of the reality. Reinforcement learning is the chosen mechanism to update the reputation value. Due to the noise underlying observations, the rule used to update the reputation value when there is a new direct interaction has a greater effect than the rule used to update the value when there is a new observation. The reputation value ranges from 0 to 1. A value greater than 0.5 represents a good performer and a value less than 0.5 represents a bad performer.

Agents can query other agents about the performance of a given partner. The answer is always a boolean value that says if the partner is good or not. In this model, liars are assumed to lie consistently, that means that every time they are queried, they return a good value for a bad target agent and vice versa. To decide, from the point of view of witness information, if a partner is good or not, the model uses the number of positive and negative answers received from witnesses. Knowing the number of witnesses and how many of them are liars, the model provides a mechanism to calculate how many agents should be queried to be sure that the likelihood of selecting a good partner has at least certain value. The subset of agents to be queried is selected randomly from the set of possible witnesses although the authors claim it is easy to add a smarter selection process based on a trust mechanism.

Because the objective of this work was to study how agents use word-of-mouth reputations to select one of several partners, agents only use witness information to take decisions. Direct experiences are only used as pieces of information to be communicated to the others. Therefore, no indication is given by the authors about how to combine direct experiences with witness information to obtain a final reputation value.

2.3.9 *AFRAS*

The main characteristic of this model [Carbo et al., 2002a] is the use of fuzzy sets to represent reputation values. Once a new fuzzy set that shows the degree of satisfaction of the latest interaction with a given partner is calculated, the old reputation value and the new satisfaction value are aggregated using a weighted aggregation. The weights of this aggregation are calculated from a single value that they call *remembrance* or *memory*. This factor allows the agent to give more importance to the latest interaction or to the old reputation value. The remembrance factor is modelled as a function of the similarity between (1) the previous reputation and the satisfaction of the last interaction and (2) the previous remembrance value. If the satisfaction of the last interaction and the reputation assigned to the partner are similar, the relevance of past experiences is

increased. If the satisfaction of the last interaction and the reputation value are different, then it is the relevance of the last experience what is increased.

The notion of reliability of the reputation value is modelled through the fuzzy sets themselves. A wide fuzzy set for a reputation value represents a high degree of uncertainty over that value while a narrow fuzzy set implies a reliable value.

Recommendations from other agents are aggregated directly with the direct experiences. The weight given to each factor (old reputation value and new opinion) is dependent on the reputation that the recommender has. Recommendations coming from a recommender with a high reputation has the same degree of reliability as a direct experience. However, opinions from an agent with bad reputation are not taken into account. To calculate the reputation of recommenders, the agent compares the recommendation with the real behaviour of the recommended agent after the interaction and increases or decreases the reputation of the recommender accordingly.

2.3.10 Carter et al.

The main idea behind the reputation model presented by Carter et al. [Carter et al., 2002] is that the reputation of an agent is based on the degree of fulfillment of roles ascribed to it by the society. All individuals have certain roles within a society. If the society judges that they have met their roles, they are rewarded with a positive reputation, otherwise they are punished with a negative reputation.

Each society has its own set of roles. As such, the reputation ascribed as a result of these roles only makes sense in the context of that particular society where roles are defined. According to this, it is impossible to universalize the calculation of reputation.

The authors formalize the set of roles within an information-sharing society and propose methods to calculate the degree of satisfaction with each of these roles. An information-sharing society is a society of agents that attempt to exchange relevant information with each other in the hopes of satisfying a user's request. They identify five roles:

- Social information provider: Users of the society should regularly contribute new knowledge about their friends to the society. This role exemplifies the degree of connectivity of an agent with its community. Each particular recommendation made by a user has a weight associated with it. This weight indicates the strength of the recommendation and is the product of a time decay factor and the reputation of the recommender. The degree to which the social information provider role is satisfied by a given user is calculated as the summation of all these weights, mapped in the interval [0,1].

- Interactivity role: Users are expected to regularly use the system. Without this participation the system becomes useless. The degree of satisfaction for this role is calculated as the number of user operations during a certain period of time divided by the total number of operations performed by all the users in the system during the same period.

- Content provider: Users should provide the society with knowledge objects that reflect their own areas of expertise. The degree of satisfaction is reflected by the

quality of the information agents that belong to that user. The quality of an agent is measured considering how close is the subject of that information agent to the user's interest. The idea is that users that create information agents related to their areas of expertise will produce higher quality content related to their interest than those who do not.

- Administrative feedback role: Users are expected to provide feedback information on the quality of the system. These qualities include easy-of-use, speed, stability, and quality of information. Users are said to satisfy this role by providing such information.

- Longevity role: Users should be encouraged to maintain a high reputation to promote the longevity of the system. The degree of satisfaction of this role is measured taking into account the average reputation of the user to the present time.

Given that, the user's overall reputation is calculated as a weighted aggregation of the degree of fulfillment of each role. The weights are entirely dependent on the specific society.

The reputation value for each agent is calculated by a centralized mechanism that monitors the system. Therefore, the reputation value of each user is a global measure shared by all the observers.

2.3.11 Castelfranchi and Falcone

The trust model proposed by Castelfranchi and Falcone [Castelfranchi and Falcone, 1998] is a clear example of a cognitive trust model. The basis of their model is the strong relation between trust and delegation. They claim that "trust is the mental background of delegation". In other words, the decision that takes an agent x to delegate a task to agent y is based on a specific set of beliefs and goals and this mental state is what we call "trust". Therefore, "only an agent with goals and beliefs can trust".

To build a mental state of trust, the basic beliefs that an agent needs are:

- Competence belief: the agent should believe that y can actually do the task.

- Dependence belief: the agent believes that y is necessary to perform the task or that it is better to rely on y to do it.

- Disposition belief: not only is necessary that y could do the task, but that it will actually do the task. In case of an intentional agent, the disposition belief must be articulated in and supported by two more beliefs:

  - Willingness belief: the agent believes that y has decided and *intends* to do α (where α is the action that allows the goal g).
  - Persistence belief: the agent believes that y is stable in its intentions of doing α.

The first two beliefs compound what they call the *core trust* and together with the disposition belief, the *reliance*. Supported and implied by the previous beliefs, another belief arises:

- Fulfillment belief: if the agent "trust in y for g", the agent decides: (i) not renouncing to goal g, (ii) not personally bringing it about, (iii) not searching for alternatives to y, and (iv) to pursue g through y.

To summarize, trust is a set of mental attitudes characterizing the "delegating" agent's mind (x) which prefers another agent (y) doing the action. y is a cognitive agent, so x believes that y *intends to do* the action and y *will persist* in this.

The degree of trust is a function of the strength of the trusting beliefs. The greater x's belief in y's competence and disposition, the greater x's trust in y. The degree of trust is used to formalize a rational basis for the decision of relying and betting on y. Also relevant is the degree of importance of the goal that is going to be achieved through the delegation. The resulting degree of trust is obtained by multiplying the degree of those beliefs and goals useful to the trust relation. If this value exceeds a given threshold and it is also the best solution of all the available solutions, then the decision to delegate is taken.

2.3.12 Summary

In table 2.1 we show a summary of the models analyzed in this review from the point of view of the classification dimensions presented in section 2.2. The abbreviations used in the table are the following:

Paradigm	N C	Numerical Cognitive
Information sources	DI DO WI SI P	Direct Interaction Direct Observation Witness Information Sociological Information Prejudice
Visibility	S G	Subjective Global
Model's Granularity	CD NCD	Context Dependent Non Context Dependent
Agent behaviour assumptions	(see section 2.2.5)	
Model Type	Trust Rep	Trust model Reputation model
General	$\times$ $\surd$ NA	No Yes Not applicable

We have to make some considerations about this table:

- We have described a set of classification aspects that allow a comparison between trust and reputation models. However, due to the diversity of such models, the classification aspects do not always fit exactly with the characteristics of the models and in some circumstances the classification for a specific model in one category or another is subjective according to our interpretation.

- We have considered only the features explicitly presented by the authors (without making suppositions on possible extensions).

- The decision of classifying the models as trust models or as reputation models is based on what the authors claim in their articles.

2.4 Trust and reputation test-beds

As mentioned previously, currently there is no test-bed that gives a common experimental environment in which we can compare computational trust and reputation models under the same conditions. Almost every model proposal is presented with a specific test-bed designed to highlight some aspects of that model. This makes the comparison between models very difficult. In this section we will present some of these test-beds. We do not intend to be exhaustive but to show a sample of the scenarios currently used to test computational trust and reputation models.

2.4.1 Test-beds based on the Prisoner's Dilemma

The prisoner's dilemma is a classic problem of game theory that raise the following situation: two people have been arrested for robbing a bank and placed in separate isolation cells. They have two options, remain silent or confess. If both remain silent, they only can be accused on firearms possession what implies a very reduced sentence. If one of them confess while the other remains silent, the one that has collaborated with the police will go free while the other will receive a big punishment. Finally, if both confess they will receive a moderate punishment.

The "dilemma" faced by the prisoners here is that, whatever the other does, it is preferable to confess than to remain silent. But the outcome obtained when both confess is worse for each than the outcome they would have obtained if they had both remained silent.

The above situation can be presented in a more formal way as follows: each player has two options, "cooperate" (C) or "defect" (D), corresponding, respectively, to the options of remaining silent or confessing in the illustrative anecdote. The payoff matrix for each action is showed in table 2.2 satisfying the following chain of inequalities: F $>$ R $>$ P $>$ S. For each possible pair of moves, the payoffs to Row and Column (in that order) are listed in the appropriate cell.

The iterated version of this game is the basis for several test-beds designed to evaluate trust and reputation models.

	Paradigm	Information sources	Visibility	Model's granularity	Agent behaviour assumptions	Boolean exchanged information?	Trust-Rep reliability measure?	Model type
S. Marsh	N	DI	S	CD	NA(5)	NA(5)	×	Trust
Online Rep. Models	N	WI	G	NCD	0	×	×	Rep
Sporas	N	WI	G	NCD	0	×	√	Rep
Histos	N	DI + WI(7)	S	NCD	0	×	×	Rep
Schillo et al.	N	$\frac{DI}{DO}$, WI	S	NCD	1	√	×	Trust
A.-Rahman and Hailes	N	DI, WI(1)	S	CD	2	4 trust values	×	Trust Rep
Esfandiary and Chandrasekharan	N	$\frac{DI}{DO}$, WI, P	S	CD	0	×	×	Trust
Yu and Singh	N	DI, WI	S	NCD	0	×	×	Trust Rep
Sen and Sajja	N	$\frac{DI}{DO}$, WI(2)	S	NCD	2(3)	√	×	Rep
AFRAS	N	DI + WI	S	NCD	2	×	√	Rep
Carter et al.	N	WI(6)	G	NCD	0	×	×	Rep
Castelfranchi and Falcone	C	NA(4)	S	CD	NA(4)	×	NA(4)	Trust
ReGreT	N	DI + WI + SI + P	S	CD	2	×	√	Trust Rep

Table 2.1: Comparison table.

(1) Direct experiences are used to compare the point of view of these witnesses with the direct perception of the agent and then be able to adjust the information coming from them accordingly.
(2) Because the objective of this work was to study how agents use word-of-mouth reputations to select one of several partners, agents only use witness information to take decisions.
(3) Liars are assumed to lie consistently.
(4) In the description of the model it is not specified how the agents obtain the information to build their beliefs.
(5) There is no exchange of information between agents
(6) Besides information coming from other users (WI) there is a central authority that monitors the agents behaviour and uses that information to build reputation.
(7) The '+' symbol means the model combines the information sources to obtain a final trust/reputation value.

	C	D
C	R,R	S,F
D	F,S	P,P

Table 2.2: Payoff matrix for the Prisoner's Dilemma

The *PlayGround*

This test-bed, designed by Marsh [Marsh, 1994] to test his trust model, consist of a 10 by 19 grid, with one cell being possibly occupied by one agent at most at any time. Agents have total freedom of movement. Interaction is achieved by a Prisoner's Dilemma played whenever and agent attempts to move into a cell which is already occupied between the occupant and the visitor. The range of vision of the agents is limited and so is the ability to move away from untrusted agents and toward trusted ones.

A *situation* is a concrete payoff matrix for a Prisoner's Dilemma game. The possible *situations* and the payoff structure for each *situation* are known by the participating agents. In any interaction, a random *situation* from a list of possible *situations* is chosen, and both agents are informed of the *situation* name. After both agents have made their decisions, fitness values are updated according to the payoff structure for the particular *situation*. Each agent also adjusts its trust values. Generally speaking, if the other cooperates, trust is increased, otherwise, trust is decreased.

Schillo et al.

Schillo et al. [Schillo et al., 2000] propose a disclosed iterated prisoner's dilemma with partner selection with a standard payoff matrix. It can be described as a five-step process:

1. Each player pays a stake.

2. Pairs of players are determined by negotiation and declaration of intentions. Agents have the possibility to deceive others about their intentions. For this step, they introduce a contract net-like protocol that is executed until each player has had the chance to find a partner.

3. The Prisoner's Dilemma game is played, bearing in mind the previously declared intentions.

4. The results are published. Due to limited perception, each agent receives only the results of a subset of all players.

5. The prizes are paid.

Agents have a limited amount of points; if an agent loses all its points, it has to retire from the game.

Mui et al.

An iterated prisoner's dilemma game where the success of your strategy gives you a greater number of descendants in the following generation is the proposal of Mui et al. [Mui et al., 2002].

The characteristics of the game are the following:

- Participants for a single game are chosen randomly from the population.
- After certain number of dyadic encounters between agents (a generation), and agent begets progeny in the next generation proportional to its fitness.
- The total population size is maintained from one generation to the next. Therefore, an increase in the number of one type of agent is balanced by a decrease in the number of other types of agents.

To the classical "always cooperate", "always defect" and "Tit for Tat" there is a new strategy that initially cooperates depending on the reputation of the other agent followed by a Tit for Tat.

Taking this test-bed as the basis, they modify several characteristics of the agents and their capabilities of interaction with the other agents in order to do experiments with different notions of reputation.

2.4.2 Castelfranchi et al.

The test-bed presented by Castelfranchi et al. [Castelfranchi et al., 1998] is designed to explore the effects of the interaction between populations following different criteria for aggression control.

The scenario of this test-bed is a grid that has randomly scattered pieces of food. The agents, that have a partial view of their environment, move through the grid in search for food, stopping to eat when they find it. Eating a piece of food takes several turns. While agents are eating they can be attacked by others. If the attack is successful, the aggressor obtains the food that the victim was trying to eat. Moving through the grid, attacking other agents or receiving attacks reduces the strength of the agent. This strength can only be recovered by eating more pieces of food. Food only reports strength to the agent once the eating process has completely finished. The goal of the agents is to increase their strength as much as possible. There are three different types of agents:

- Blind agent. Aggression is constrained only by personal utility.
- Strategic agent. Aggression is performed only if it knows it will win the contest.
- Normative agent. In this case, possession of food is ascribed to an agent on the grounds of spatial vicinity. This type of agent follows the norm that says that agents cannot attack possessors eating their ascribed food.

In further experiments the authors extend the capabilities of normative agents by adding the notion of reputation and the possibility of exchanging information about cheaters.

2.4.3 SPORAS, ReGreT and AFRAS

The set of experiments presented in this section where first used by Zacharia et al. [Zacharia, 1999] to test the Sporas model. The same set of experiments where used by Sabater and Sierra [Sabater and Sierra, 2001] to compare the ReGreT model with Sporas and the reputation mechanism used in Amazon Auctions. Finally, an extended version of these experiments was proposed by Carbo et al. [Carbo et al., 2002b, Carbo et al., 2002a] to compare AFRAS § 2.3.9 with Sporas § 2.3.3, ReGreT, Yu and Singh's § 2.3.7 model and two online reputation mechanisms (eBay and Bizrate) § 2.3.2. It has to be clear that these sets of experiments are far from a real test-bed because they focus on a restricted set of situations. However, together with the prisoner's dilemma, they are the only set of experiments used by several authors to compare their reputation models under the same conditions.

The initial set of experiments focus on the convergence speed and the abuse of prior performance. The objective is to analyze how the models react to the different scenarios.

The experiment to test the convergence speed proposes a scenario with a fixed number of users with uniformly distributed real reputations. The users start with minimum reputation. In each period of the simulation, the users are matched randomly and get rated by each other according to their actual performance. The objective is to see how long it takes for the models to reach the real reputation value.

In the abuse of prior performance scenario, a user joins the marketplace, behaves reliably until s/he reaches a high reputation value and then starts abusing his/her reputation to commit fraud. The aspect to be analyzed here is how quickly the models can adapt to the new situation.

Carbo et al. [Carbo et al., 2002b, Carbo et al., 2002a] extend the set of experiments with studies about how the use of cooperation between agents improve the convergence of reputation values and the impact of coalitions between sellers and buyers.

2.5 Conclusions

After the analysis performed in this chapter, there are several things to be considered.

If we observe table 2.1, it is clear that numerical modelling is the predominant paradigm used nowadays for the design of computational trust and reputation models. Possibly, the reason for that is the profile of people that is working in the area of multi-agent systems and e-commerce (economists and computer scientist) with a strong background in game theory and AI techniques.

As would be expected, the main sources of information used by the trust and reputation models are direct experiences and information from third party agents (witness information). There are very few models (in fact only the ReGreT system and the model proposed by Esfandiary and Chandrasekharan) that take into account other aspects to calculate trust and reputation values.

We think that a good mechanism to increase the efficiency of actual trust and reputation models (and also to overcome the lack of confidence in e-markets) is the introduction of sociological aspects as part of these models. It is true that in the actual e-markets

this kind of sociological information is almost inexistent or it is not available to the participating agents. Therefore, nowadays, a model that uses this kind of information is not more useful than simpler models that only take into account (for example) direct experiences. It has no sense to increase the complexity of trust and reputation models if later on you have to use them in an environment where it is not possible to exploit their capabilities.

Does it mean we have to renounce to sophisticated trust and reputation models? Certainly not. Electronic societies have to evolve to a new stage of complexity where interaction and links between their members become more relevant (in a similar way —taking into account the differences— we found in human societies).

Coming back again to table 2.1, we see that only the ReGreT system, the AFRAS model and, in a way, the Histos model, propose methods to combine different sources of information. Also, it is not usual to provide with the trust and reputation values a measure of their reliability, something that we think it is very important for the agent that has to use those values.

Finally, we observe that models are oriented to trust or to reputation but it is not common to have both integrated in the same system.

With respect to test-beds, we think it is time to propose a set of benchmarks that allow the comparison of the different reputation models that currently exist. The number of reputation models is increasing quickly and we need an objective way to compare the different approaches, their benefits and drawbacks.

Chapter 3

The *SuppWorld* framework

3.1 Introduction

Although there are several frameworks that can be used to test computational trust and reputation models (see chapter 2), none of these frameworks provides a scenario rich enough to test all the dimensions of the ReGreT system. The main drawback in these frameworks is that they are not prepared to naturally support social relations among members. This is because they are designed to test trust and reputation models that do not consider this aspect. As we will see, social relations play an essential role in the ReGreT system. Therefore, we decided to design and implement a new framework adapted to the special characteristics of ReGreT. In this chapter we present *SuppWorld*, the framework we have used to test the ReGreT system and perform the experiments detailed in chapter 7. This framework allows us to build simple scenarios oriented to test the basic aspects of trust and reputation models and also complex scenarios where social relations acquire a great relevance.

3.2 The *SuppWorld* framework, an overview

The *SuppWorld* is built around the idea of a supply chain. In a typical supply chain, agents trade by buying one level below in the chain, adding value to the purchased goods, and selling the manufactured good up to the next level in the chain. The *SuppWorld* allows the design of scenarios based on this structure and oriented to the study of negotiation, trust and reputation models.

The activity in a *SuppWorld* scenario is organized in several scenes that can be combined to recreate different supply chain configurations. These scenes are:

- Markets. As the name suggests, in these scenes is where agents trade. A set of sellers sell one or different types of products to buyers. Agreements between sellers and buyers materialize after a negotiation process.
- Conventions. Here is where an agent can exchange information with other agents of the same type. It is also the place where agents can establish coalitions.

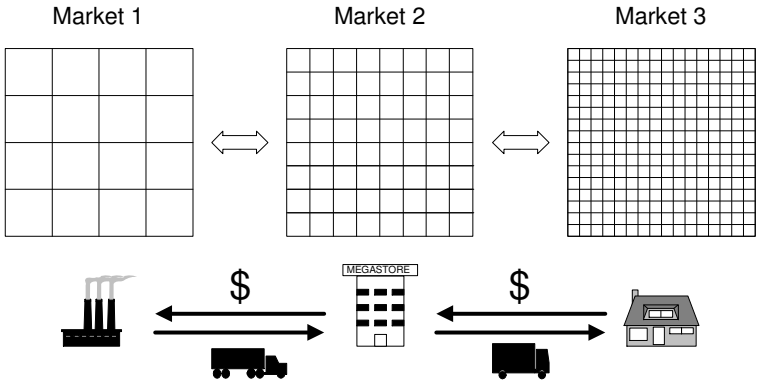

Figure 3.1: *SuppWorld* example scenario.

- Entrance of rough material. A supply chain is not a closed environment. It is necessary to allow the entrance of rough material that, properly transformed, will follow its path up to the final consumers.

- Production process. In each link of the supply chain there is a process of transformation that adds new value to the goods. This process is simulated in this scene.

- Entrance of money. Similarly to the entrance of rough material, it is necessary at least one point to introduce money to the supply chain. With this scene you can simulate the action of final consumers or a "pay day" for these final consumers if they are explicitly represented.

The heartbeat of a *SuppWorld* scenario is measured in 'ticks'. All the activities performed during a tick, from the point of view of the simulation, are supposed to run in parallel.

In the following sections we describe in detail each scene type.

3.3 Markets

The market is the main scene in a *SuppWorld* scenario. A market is represented as a toroidal grid where each cell is owned by a single seller. In a *SuppWorld* scenario you can have several markets each one with their own sellers. As we have said, the markets are organized as a supply chain, that is, the buyers in the first market are the sellers in the second market, the buyers in the second market are the sellers in the third one and so on. From now on, we will refer to the *home* grid of an agent as the grid (market) where this agent acts as a seller and, therefore, where it has its home cell. Figure 3.1 shows a typical configuration of *SuppWorld* markets.

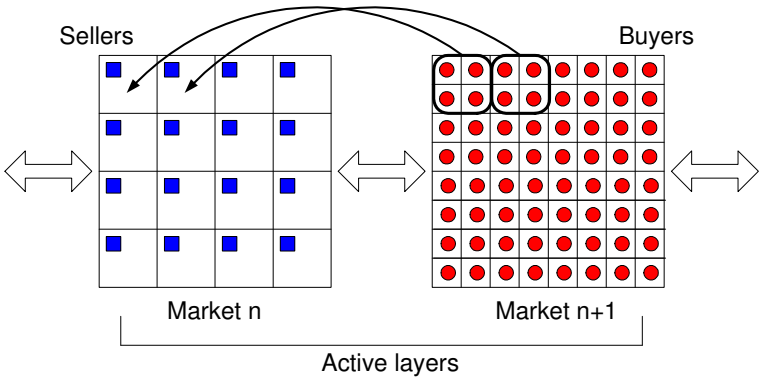

Figure 3.2: Fixing the initial position of the buyers in the market.

Markets are opened one at a time and each market remains opened during a configurable amount of time. This period that the market is open is what we call a *market session*.

The physical location of the sellers in the home grid determines the position from which they will start in the market when they act as buyers. There is a function that distributes uniformly the buyers over the active market grid at the beginning of each market session taking into account their position in the home grid (see Figure 3.2).

As we have said, the interaction of a buyer with the seller to buy goods is through a negotiation process. This process is explained in section 3.8. For the moment it is enough to know that both parts can withdraw from the negotiation at any moment. If the negotiation leads to an agreement between the seller and the buyer, the seller sends the goods to the buyer's home cell (to be exact, to the mapped position of the buyer's home cell into the current grid). The buyer will pay the goods as soon as they arrive to its destination but before it can inspect the delivery. Of course, both the seller and the buyer are free to decide how they are going to fulfill the contract. Because the main goal of this framework is the study of trust and reputation, there are no punishment mechanisms for those agents that commit fraud and cheat the others. We want to test the reaction of agents endowed with trust and reputation mechanisms.

While sellers stay always in its cell during a market session, buyers can move freely from one cell to another. Buyers can move only to one of the adjacent cells at a time (including diagonal movements) and this movement has a cost in time units (ticks). There is a single exception to this. After the negotiation process, the seller can recommend another seller to the buyer. If the buyer decides to follow the recommendation, it can move directly from the actual cell to the cell of the recommended seller. The cost in time for that movement is substantially less than moving to the same location without following the recommendation. In other words, following a recommendation is like a "direct flight" to the recommended cell. Giving a recommendation after the negotiation process is not mandatory for the seller.

Because it is possible to have more than one buyer in each cell (in fact this will be

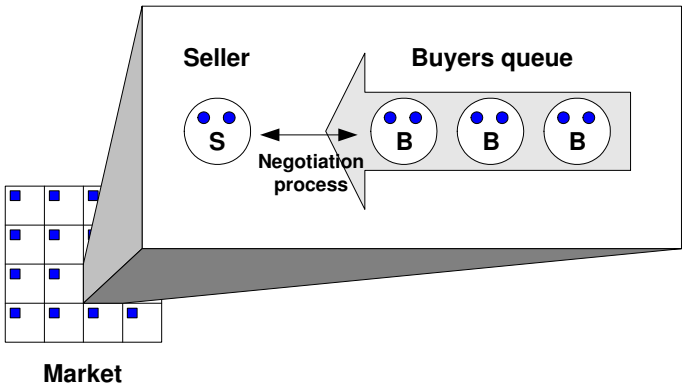

Figure 3.3: Cell structure.

usually the case), the cells have a local queue where the buyers can wait their turn to interact with the seller. If several buyers arrive at the same time on the same cell they are added to the queue randomly. Only the first in the queue can interact with the seller. The other buyers in the queue have three options: just waiting for its turn to interact with the seller, move to another cell, or query one of the other buyers that are waiting in the same queue.

The buyer that has interacted with the seller cannot stay in the same cell after the negotiation and has to move necessarily to one of the adjacent cells or follow the recommendation (if it was performed).

Figure 3.3 shows the structure of a generic cell and figure 3.4 the activity flow in a cell during a tick.

The first action is the exchange of information among buyers that are waiting in the queue. This exchange of information is performed in turns and each agent has a fixed number of opportunities to query the others. Then, the first buyer negotiates with the seller and the seller gives (or not) a recommendation. The third stage gives the agents the possibility to explore the nearest environment. Each agent receives the following information about the adjacent cells: Who is the owner of the cell, the type of product sold in that cell and the name of the agents waiting in its queue. This information can be used by the agents in the final stage in order to decide if it is worth it to move to another cell and, in that case, which is the best movement.

3.4 Conventions

The convention scene has two main functions in a *SuppWorld* scenario. First, it allows agents hosted in the same grid to query their partners about other agents' reputation. Conventions give the possibility to query specific agents instead of relying on casual meetings in the queue of a cell. This is specially important for agents in the first market of a *SuppWorld* scenario. Because the agents in the first market never take the role of

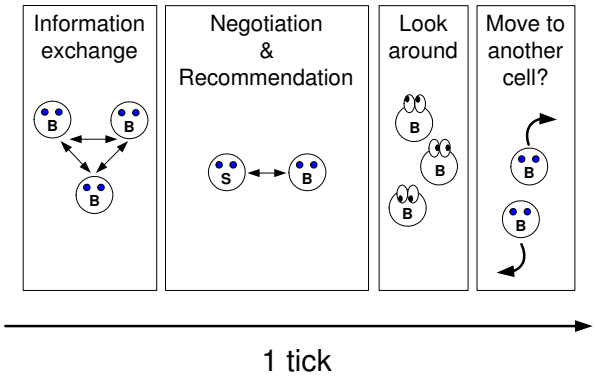

Figure 3.4: Activity flow in a generic cell.

buyers, they cannot meet in the queue of a cell to exchange information as the others do. As in the waiting queue in a cell, the exchange of information is performed in turns and each agent has a limited number of opportunities to query the others.

The second purpose of conventions is to give the agents the possibility of establishing coalitions. As we have said, sellers can recommend other sellers after a negotiation process. The (rational) reason for a seller a to recommend a seller b is based on the belief that b will correspond recommending a to its clients. This belief is not an act of faith but an act based on a previous established coalition between both agents in a convention scene. It is important to note that a seller knows those agents that are in its cell because of a recommendation and who was the recommender. This allows sellers to evaluate the benefits of a coalition and if it is worth maintaining it.

3.5 Entrance of rough material

Every supply chain needs at least one point to allow the entrance of new material. This is the purpose of this scene. The new material usually enters the supply chain in the first layer. The amount of new material is fixed as a parameter of the scenario. Each agent has a storage capacity and it has to cope with all the material assigned to it. If it has not enough money or there is no free space in the store to put the new material, the material is lost and the agent has to pay a penalty quote for each lost unit. Therefore, an agent that cannot trade and move away the material present in its stores is condemned to bankruptcy.

3.6 Production process

The production process recreates the transformation of goods produced in each layer of a supply chain. This transformation adds new value to the original product. Then,

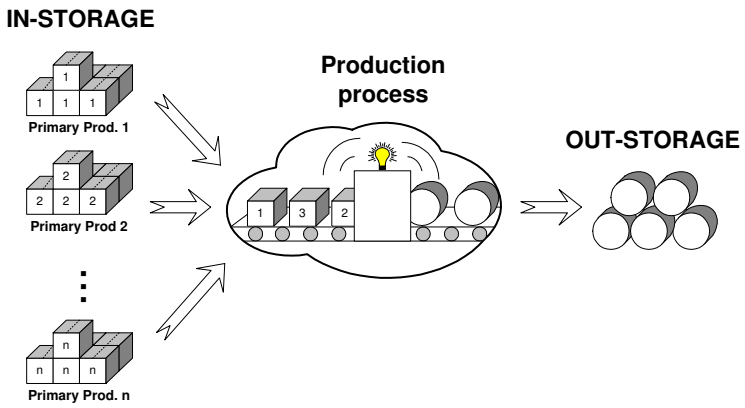

Figure 3.5: *SuppWorld* agent's storage facilities.

the transformed product can be sold to agents in the next layer with the corresponding benefit for the seller.

Figure 3.5 shows the storage facilities of a *SuppWorld* agent.

A *SuppWorld* agent has a set of in-storage facilities (one for each primary product necessary to generate the final product) and one out-storage facility (where the final product is stored until it can be sold in the next layer). To generate one unit of final product, an agent needs one unit of each primary product. During the market sessions (or during the entrance of rough material in the case of agents belonging to the first layer of the supply chain) buyers acquire primary products. Each unit of primary product has an associated quality. This quality will determine the quality of the final product and, therefore, its value in the market. The best way to understand how the quality of primary products determine the quality of the final product is through an example.

Suppose the initial situation just before the production process showed in Figure 3.6(a). This agent needs one unit of primary product A and one unit of primary product B to generate one unit of final product AB. It has a certain quantity of product A with quality 1 (the worst possible quality) that needs to be mixed with product B. Although the worst quality of product B available is 2, the mix of quality 1 and quality 2 generates final product of the minimum quality, that is, quality 1. The same process is applied to each layer of quality as shown in Figure 3.6(b,c,d,e,f). At the end, the primary product that cannot be transformed to final product because there is a lack of other primary products, is lost (you can think we are dealing with perishable products like food).

Generating each unit of final product is not free. There is a cost associated to each generated unit. Moreover, the out-storage facility has a finite capacity. Either if the agent cannot pay the production cost or there is not enough space in the out-storage facility to store the final product, the final product cannot be generated and the primary products are lost.

The final result of the production process is a certain amount of final product in the

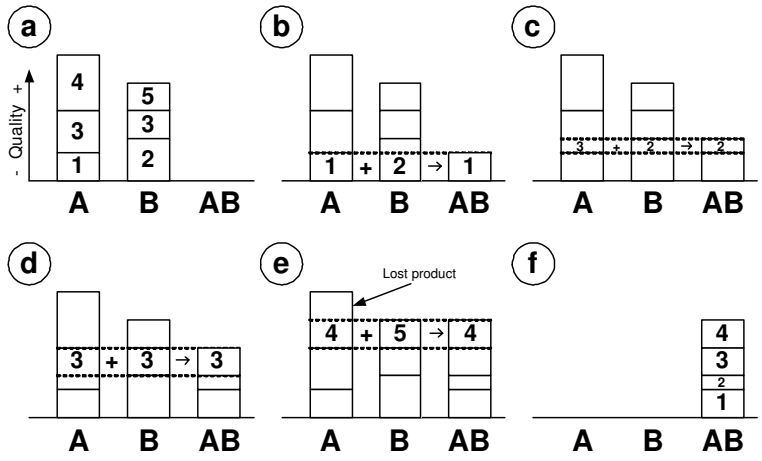

Figure 3.6: Production process example.

out-storage facility, with different qualities and ready to be sold in the next layer of the supply chain.

3.7 Entrance of money

At the end of the supply chain we have the final consumers. They are responsible of adding money to the chain. In a *SuppWorld* scenario, this is modelled using the 'entrance of money' scene. This scene allows two possibilities. The first possibility is to simulate a layer of final consumers that buy everything from the last sellers in the chain. The price that is paid for the products is proportional to their quality. The second possibility is the simulation of a salary for the final consumers. This is necessary if we make explicit the final consumers layer.

3.8 Negotiation process

The mechanism that the agents use to buy and sell products in a *Suppworld* scenario is a one-to-one negotiation. The negotiation model we have implemented in our prototype is the one presented by Peyman Faratin, Carles Sierra and Nick R. Jennings in [Faratin et al., 1997]. It is a model for bilateral negotiations about a set of quantitative variables. In the case of a *SuppWorld* scenario, this variables are the price, the quantity, the quality and the transport used by the seller to deliver the product.

Offers and counter offers are generated by linear combinations of simple functions, called *tactics*. Tactics generate an offer, or counter offer, for a single component of the negotiation object (in our case, price, quantity, quality and transport type) using a single criterion (time, resource, etc.). Different weights in the linear combination allow

the varying importance of the criteria to be modelled.

There are two tactics available to agents in a *SuppWorld* scenario: time dependent tactics and tit-for-tat. A time dependent tactic uses time as the predominant factor to decide which value to offer next. The acceptance value for the issue depends on the remaining negotiation time. On the other hand, tit-for-tat tactics compute the next offer based on the previous attitude of the negotiation opponent.

A *strategy* determines how the agent changes the linear combination of tactics over time. For instance, the agent can start using a tit-for-tat tactic and change gradually to a time dependent tactic when the negotiation deadline is approaching.

3.9 The agents behaviour

There are two elements in a *SuppWorld* scenario that determine the behaviour of an agent.

The first is what we call the *alignment* of the agent. The *alignment* defines the basic behaviour of the agent in aspects like how contracts will be fulfilled or the truthfulness of the information given to other members of the society.

The second are the social relations between the agent and the other members of the society. In a *SuppWorld* scenario there are three types of social relations among their members:

- *Competition*. This is the type of relation found between two agents that pursue the same goals and need the same (usually scarce) resources. In this situation, agents tend to use all the available mechanisms to take some advantage over their competitors, for instance hiding information or lying. A competitive relation between and agent a and an agent b is noted as *comp(a,b)*.

- *Cooperation*. This relation implies significant exchange of sincere information between the agents and some kind of predisposition to help each other if possible. Notice that we talk about "sincere" information instead of "true" information because the agent who gives the information *believes* it is true. We consider that two agents cannot have at the same time a competitive and a cooperative relation. This is also the relation type that identifies groups of agents. A cooperative relation between and agent a and an agent b is noted as *coop(a,b)*.

- *Trade* (*trd*). This type of relation reflects the existence of commercial transactions between two agents and is compatible either with cooperative or competitive relations. For the moment this is the only social relation that agents can identify by themselves in a *SuppWorld* scenario. A trade relation between and agent a and an agent b is noted as *trd(a,b)*.

The social relations can change the basic behaviour defined by the *alignment*. For instance, an agent that normally provides false information to the others, will tell the truth to an agent that has a cooperative relation with it.

3.10 Implementation

The *SuppWorld* framework is not a real multi-agent system. Agents are implemented as C++ classes and everything is synchronous and sequential. We decided to use this approach instead of a real multi-agent system (with agents running in parallel exchanging messages through a communication platform) because it gives us more control over the execution.

An experiment is specified using a text file with several blocks that need to be defined:

- General parameters: The general parameters of the experiment like the number of rounds and the sequence of scenes in each round.

- Transport: A list of the different transport types that are available to the sellers in order to send the product to the buyers. Each transport type has a cost and a speed associated.

- Product profiles: The different products and their characteristics. Production cost and price range are examples of the parameters that define a product.

- Negotiation and trust/reputation engines: The specification of the negotiation and trust/reputation engines that will use the agents in that experiment. The framework allows the use of a different engine (or the same engine with different parameters) for each class of agent. The *SuppWorld* framework has been designed not only to test the ReGreT system but trust and reputation models in general. We have put special effort in the connexion between the kernel of the agent and the trust and reputation model in order to facilitate the addition of other trust and reputation models.

- Individuals: The different types of individuals that will populate the *SuppWorld* scenario. The products each type of agent is interested in or the type of trust and reputation model they use are the kind of agent properties defined here.

- Grids: The composition of the different markets. Once you have fixed the dimensions of the grid, you have several options to specify the agents that will populate it. For each cell in a grid, it is possible to specify the type of the agent (the owner of the cell) individually or let the program to choose an agent type randomly from a list of possibilities. Similarly you can choose the behaviour of the agent or define a distribution of behaviours to be applied automatically.

- Conventions: The definition of conventions. It is limited to the grid and the number of ticks given to agents in that grid to exchange information.

- Societies: The initial set of cooperative and competitive relations available to each agent. As we have said, at this moment only *trade* relations can be detected by agents. Therefore we need a mechanism to define the view agents will have about cooperative and competitive relations.

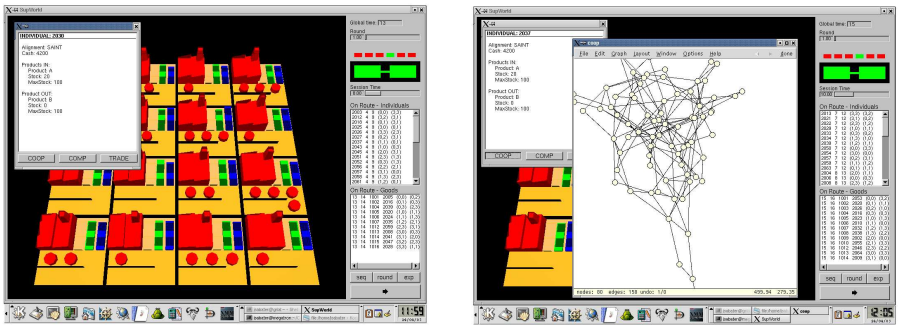

Figure 3.7: The *SuppWorld* graphical mode.

The framework has a graphical mode that allows users to monitor the experiment during the execution and a silent mode to run experiments in the background. Figure 3.7 show two screen shots of the *SuppWorld* framework running in graphical mode. The framework also has a log system to register the evolution of the experiments.

In appendix B you can find a detailed description of the configuration files and some examples corresponding to the experiments presented in chapter 7.

Chapter 4

The ReGreT system

4.1 Introduction

Up to now, the computational models of trust and reputation have been considering two different information sources: (i) the direct interactions among agents and (ii) the information provided by members of the society about experiences they had in the past [Sabater and Sierra, 2001, Schillo et al., 2000, Yu and Singh, 2000, Zacharia, 1999]. Those systems, however, forget a third source of information that can be very useful. As a direct consequence of the interactions, it is possible (even in not too complex societies) to identify different types of social relations between society members. Sociologists and psychologists have been studying these social networks in human societies for a long time and also how these social networks can be used to analyse trust and reputation [Pujol et al., 2003, Buskens, 1999]. These studies show that it is possible to say a lot about the behaviour of individuals using the information obtained from the analysis of their social network.

In this chapter we present ReGreT, a modular trust and reputation model oriented to complex e-commerce environments where social relations play an important role.

The main characteristics of ReGreT are:

- It takes into account direct experiences, information from third party agents and social structures to calculate trust, reputation and credibility values.
- It has a trust model based on direct experiences and reputation.
- It incorporates an advanced reputation model that works with transmitted and social knowledge.
- It has a credibility module to evaluate the truthfulness of information received from third party agents.
- It uses social network analysis (see section 4.2) to improve the knowledge about the surrounding society (specially when no direct experiences are available).

- It provides a degree of reliability for the trust, reputation and credibility values that helps the agent to decide if it is sensible or not to use them in the agent's decision making process.

- It can adapt to situations of partial information and improve gradually its accuracy when new information becomes available.

- It can manage at the same time different trust and reputation values associated to different behavioural aspects. Also it can combine reputation and trust values linked to simple aspects in order to calculate values associated to more complex attributes.

Some people could argue that current e-commerce scenarios are not complex enough to justify the degree of complexity modelled by the ReGreT system. Although, as we will see, the modular design of the ReGreT system makes possible its use in a wide range of environments, it is true that in order to fully exploit its capabilities (specially those related with social network analysis) it is necessary a certain degree of complexity in the society. We agree that this complexity is not present in current operative e-commerce environments but we take the stance that in the near future, as the complexity of tasks to be performed by agents will increase, these kind of complex scenarios will become usual.

In the next section we will make a short presentation of what social network analysis is and why we think it can be used as part of trust and reputation models to increase their performance when used in complex societies. After a general perspective of the ReGreT system in section 4.3 we explain the different elements that compound it. We start with the direct trust module § 4.4. Then we analyse the different parts of the reputation model § 4.5 (witness reputation § 4.5.1, neighbourhood reputation § 4.5.2 and system reputation § 4.5.3) and how they are combined to become a complete reputation mechanism § 4.5.4. The credibility module is explained in section 4.5.1 and the trust model in section 4.6. To finish the presentation of the ReGreT system, we go through the ontological dimension in section 4.7.

4.2 Social Network Analysis and agent societies

Social network analysis is the mapping and measuring of relationships between people, groups, organizations, computers or other information/knowledge processing entities. The nodes in the network are the people and groups while the links show relationships between nodes. Social network analysis provides both a visual and a mathematical analysis of these relationships.

As pointed out by Scott [Scott, 2000], three main traditions have contributed to the development of present-day social network analysis: the advances on graph theory performed by the sociometric analysts; the Harvard researchers of the 1930s, who explored patterns of interpersonal relations and the formation of 'cliques'; and the Manchester anthropologists, who built on both of these strands to investigate the structure of 'community' relations in tribal and village societies. These traditions were brought together

in the 1960s and 1970s to forge contemporary social network analysis. From then, social network analysis has been widely used in the social and behavioral sciences, as well as areas like political science, economics, or industrial engineering.

One of the main characteristics of social network analysis is the use of relational data instead of attribute data (which is usually quantified and analysed through statistical methods). Relational data can be handled and managed in matrix form or using graphs. A graph structure that shows social relations is called a *sociogram*. A different sociogram is usually built for each social relation under study and depending on the type of relation we have a directed or non-directed sociogram, with weighted edges or without. Indegree, density or node centrality are examples of graph theory concepts used in social network analysis to extract conclusions from sociograms.

We know that social network analysis can be used to analyze human societies, but could it also be suitable for agent societies? The greater simplicity, in social terms, of multi-agent systems suggests that social network analysis could be applied to autonomous agents with even better results. Of course, social network analysis can be useful in agent communities with a certain degree of complexity. It has no sense the use of these techniques in simple societies where it is not possible to establish relations of any kind. Also, we have to assume that agents are rational and that normally behave according to these relations.

We have exposed the pros but, of course, there is a con. Obviously, the more relational data the better the network analysis is. However, these data can be difficult to obtain. Sociologists usually get them through public-opinion polls and interviews with the individuals. This procedure is, a priory, not possible in agent societies. Moreover, the analysis of human social structures is usually done by a sociologist external to the society. This external position gives the analyst a privileged watchtower to make this kind of analysis. In our case, as we want to use social analysis as part of the decision making mechanism of the agent (specifically as part of the reputation and credibility models), each agent has to do this analysis from its own limited vision of the world.

It is beyond the scope of the present work to propose solutions about the way an agent builds such sociograms. We will assume that the agent owns a set of sociograms that show the social relations in its environment. These sociograms are not necessarily complete or accurate. We suppose they are built by each agent using the knowledge it has about the environment. Therefore, sociograms are dynamic and agent dependent.

Before finishing this short presentation of social network analysis we want to mention two works closely related with the one we are presenting here. Both works use social network analysis, the first as part of a trust model and the second as part of a reputation model.

In his work, Buskens [Buskens, 1998, Buskens, 1999] focus on how trust depends on different structural properties of social networks such as density, outdegree, indegree and centralization. Individual and global network parameters are distinguished to explain learning effects and control effects through social networks of trustors. Individual network parameters refer to the network connections of the individual with the rest of the net (indegree, outdegree, etc.) while global parameters refer to the network as a whole (density, centralization, etc.). Individual network parameters are used to explain differences in trust within one network while global network parameters are useful to

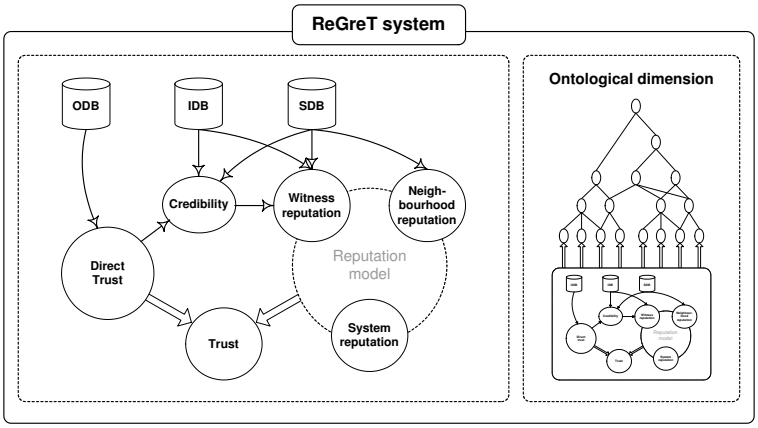

Figure 4.1: The ReGreT system.

explain differences in trust between networks.

Pujol et al. [Pujol et al., 2002, Pujol et al., 2003] propose a reputation mechanism based on the position of each member of the community within the corresponding social network. They propose *NodeRanking*, an algorithm inspired in the ranking algorithms for web pages based on web topology. The main idea is that each node has an authority and part of this authority is propagated to the out-nodes via out-edges. At the same time, the authority of a node depends on the authority of its in-nodes.

As we will see, the ReGreT system uses social network analysis in two different situations. One is to choose a good set of witnesses to be queried for information. The way social network analysis is used here has a lot of aspects in common with the way it is used in the work of Buskens and Pujol et al. In this situation it is considered only one type of relation and the analysis is based on parameters like centrality, the number of other points in its neighbourhood (degree) and so on. However, in the ReGreT system, social network analysis is also used as part of the reputation and credibility models. In both cases only the relations among a small set of individuals are considered and the type of relation is very relevant in order to perform the analysis. Having more than one type of relation and how the relation types are taken into account for the analysis differentiates ReGreT from the models proposed by Pujol and Buskens.

4.3 The ReGreT system, a general view

Figure 4.1 shows a panoramic view of the ReGreT system.

The system maintains three knowledge bases. The outcomes data base (*ODB*) to store previous contracts and their result; the information data base (*IDB*), that is used as a container for the information received from other partners and finally the sociograms data base (*SDB*) to store the sociograms that define the agent social view of the world. These data bases feed the different modules of the system.

The *direct trust* module deals with direct experiences and how these experiences can contribute to the trust on third party agents. Together with the reputation model they are the basis for the trust model.

The reputation model is divided in three specialized types of reputation depending on the information source that is used to calculate them. If the reputation is calculated from the information coming from witnesses we talk about the *witness reputation*, if the reputation is calculated using the information extracted from the social relations between partners we are talking about the *neighbourhood reputation*. Finally, reputation based on roles and general properties is modelled by the *system reputation*.

The system also incorporates a credibility module that allows the agent to measure the reliability of witnesses and their information. As we will see, this module is extensively used in the calculation of *witness reputation*.

All these modules work together to offer a complete trust model based on direct knowledge and reputation. However, the modular approach in the design of the system allows the agent to decide which parts it wants to use. For instance, the agent can decide not to use *neighbourhood reputation* to calculate a reputation value or rely only on *direct trust* to calculate the trust on an agent without using the reputation module.

Another advantage of this modular approach is the adaptability that the system has to different degrees of knowledge. As we will see, the system is operative even when the agent is a newcomer and it has an important lack of information. As long as the agent increases its knowledge about the other members of the community and the social relations between them, the system starts using other modules to improve the accuracy of the trust and reputation values. This allows the system to be used in a wide range of scenarios, from the most simple to the most complex. If the information is available, the system will use it.

In the ReGreT system, each trust and reputation value has an associated reliability measure. This measure tells the agent how confident the system is on that value according to how it has been calculated. Thanks to this measure, the agent can decide, for example, if it is sensible or not to use the trust and reputation values as part of the decision making mechanism.

The last element in the ReGreT system is the *ontological structure*. As we argued in the introduction, we consider that trust and reputation are not single and abstract concepts but rather multi-facet concepts. The *ontological structure* provides the necessary information to combine reputation and trust values linked to simple aspects in order to calculate values associated to more complex attributes. For example, the reputation of being a good flying company summarizes the reputation of having good planes, the reputation of never losing luggage and the reputation of serving good food. In turn, the reputation of having good planes is a summary of the reputation of having a good maintenance service and the reputation of frequently renewing the fleet. Note that each individual can have a different *ontological structure* to combine trust and reputation values and a different way to weigh the importance of these values when they are combined.

Trust and reputation have a temporal dimension. That is, the reputation and trust value of an agent change along time. We will, however, omit the reference to time in the notation in order to make it more readable. We will refer to the agent that is calculating

a reputation as a (what we call the "source agent") and the agent that is the object of this calculation as b (what we call the "target agent").

In following sections we will describe in detail each one of the elements that compound the ReGreT system.

4.4 Direct trust

We use the term *direct trust* to refer to the trust that is build from direct interactions (that is, using information perceived by the agent itself) in comparison to general *trust* that is build using also other elements like reputation that depends on opinions and observations of third party agents (see section 4.6).

For simplicity we assume that there is no difference between direct interaction and direct observation in terms of reliability of the information so, from now on, we will talk about direct experiences to refer to both types of information. In the ReGreT system, *direct trust* is always linked to a specific behavioural aspect. Therefore, we talk about the *direct trust* agent a has in agent b in a specific context to perform a specific action. I can trust a friend to drive me to the airport but it doesn't mean I trust him to fly the plane. The ReGreT system, either for trust or reputation, always takes into account the context.

The basic element to calculate a *direct trust* in the ReGreT system is the *outcome*. We define the *outcome* of a dialog between two agents as either:

- An initial contract to take a particular course of action and the actual result of the actions taken, or
- An initial contract to fix the terms and conditions of a transaction and the actual values of the terms of the transaction.

An outcome is represented as a tuple of the form $o = (a, b, I, X^c, X^f, t)$ where a and b are the agents involved in the contract, I a set of indexes that identify the issues of the contract, X^c and X^f are two vectors with the agreed values of the contract and the actual values after its fulfillment respectively, and t the time when the contract was signed. We use a subscript $i \in I$ to refer to the specific value of issue i in vectors X^c and X^f. For instance, in a SuppWorld scenario we have $I = \{Price, Quantity, Quality, Transport\_Type\}$. If we want to make reference to the *Price* value in the vector X^c we use the notation X^c_{Price}.

ODB is defined as the set of all possible outcomes. $ODB^{a,b} \subseteq ODB$ is the set of outcomes that agent a has signed with agent b. We define $ODB^{a,b}_{\{i_1,\cdots,i_n\}} \subseteq ODB^{a,b}$ as the set of outcomes that include $\{i_1, \cdots, i_n\}$ as issues in the contract. For example, $ODB^{a,b}_{\{Price\}}$ is the set of outcomes that has agent a from previous interactions with agent b and that fix, at least, the value for the issue *Price*.

Given that, we can define a *direct trust* (noted as $DT_{a\to b}(\varphi)$ where φ is the behavioural aspect under evaluation) as the trust relationship calculated directly from an agent's outcomes database.

To calculate a *direct trust* relationship we use a weighted mean of the outcomes evaluation, giving more relevance to recent outcomes.[1] The evaluation of an outcome $o = (a, b, I, X^c, X^f, t)$ (what we call the *impression* of the outcome) depends on the behavioural aspect. This dependency is reflected in two aspects. First, the issue of the outcome that is relevant for the evaluation and second the function used for the evaluation.

We define a *grounding relation* (*gr*) as the relation that links a behavioural aspect φ with a specific issue and the function used to evaluate the outcome. This allows us to select the right subset of outcomes from the general outcomes' data base and also evaluate the outcome according to the semantics of the behavioural aspect.

As an example, a possible *grounding relation* for a seller in a SuppWorld scenario is defined in the following table:

φ	$gr(\varphi)$	$V(X^s) \otimes V(X^c)$
offers_good_prices	*Price*	$V(X^s) - V(X^c)$
maintains_agreed_quantities	*Quantity*	$abs(V(X^s) - V(X^c))$
offers_good_quality	*Quality*	$V(X^s) - V(X^c)$
delivers_quickly	*Transport_Type*	$V(X^s) - V(X^c)$

where $V(X^c)$ is the utility of the contract values, and $V(X^s)$ is the utility of a vector build using the following formula:

$$X_i^s = \begin{cases} X_i^f & \text{if } i \in gr(\varphi) \\ X_i^c & otherwise \end{cases}$$

In other words, we obtain this vector from vector X^c by replacing the value specified in the index $gr(\varphi)$ by the value in the same positions in vector X^f.

The general formula to evaluate an outcome is:

$$Imp(o, \varphi) = g(V(X^s) \otimes V(X^c))$$

Where g is a function that models the personality of the agent as the degree of deception or reward obtained after the analysis of the outcome (an appropriate function is $g(x) = \sin(\frac{\pi}{2}x)$ shown in figure 4.2) and $\otimes$ is an aggregation function that depends on the shape of the utility function for that issue. For instance, if instead of having the behavioural aspect *offers_good_prices* we had *offers_bad_prices*, the function $\otimes$ would be $V(X^c) - V(X^s)$.

Given that, the formula to calculate a *direct trust* value in the ReGreT system is:

$$DT_{a \to b}(\varphi) = \sum_{o_i \in ODB^{a,b}_{gr(\varphi)}} \rho(t, t_i) \cdot Imp(o_i, \varphi)$$

with $\rho(t, t_i) = \frac{f(t_i, t)}{\sum_{o_j \in ODB^{a,b}_{gr(\varphi)}} f(t_j, t)}$ where t is the current time and $f(t_i, t)$ is a time dependent function that gives higher values to values closer to t. A simple example of this type of function is $f(t_i, t) = \frac{t_i}{t}$.

[1] There are many psychological studies that support recency as a determinant factor [Karlins and I.Abelson, 1970].

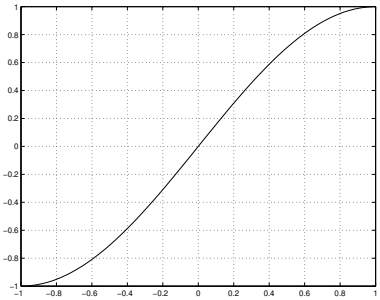

Figure 4.2: $g(x) = \sin(\frac{\pi}{2}x)$

We know how to calculate a *direct trust* value. However in order to use that value it is very important for the agent to know also how reliable it is. There are many elements that can be taken into account to calculate the reliability of a *direct trust* value. The ReGreT system focus on two of them: the number of outcomes used to calculate the *direct trust* value and the variability of their values. This approach is similar to that used in the Sporas reputation model [Zacharia, 1999].

The intuition behind the number of outcomes factor (noted as *No*) is that an isolated experience (or a few of them) is not enough to make a correct judgment about somebody. You need a certain amount of experiences before you can assess how an agent behaviour is. As the number of outcomes grows, the reliability degree increases until it reaches a maximum value, what we call the *intimate* level of interactions (*itm* from now on). From a social point of view, this stage is what we know as a close relation. More experiences will not increase the reliability of our opinion from then on. The next simple function is the one we use to model this:

$$No(ODB^{a,b}_{gr(\varphi)}) = \begin{cases} \sin\left(\frac{\pi \cdot |ODB^{a,b}_{gr(\varphi)}|}{2 \cdot itm}\right) & |ODB^{a,b}_{gr(\varphi)}| \leq \text{itm} \\ 1 & \text{otherwise} \end{cases}$$

The function chosen to compute $ODB^{a,b}_{gr(\varphi)}$ when $|ODB^{a,b}_{gr(\varphi)}| \leq$ itm serves the purpose of reaching the value 1 when $|ODB^{a,b}_{gr(\varphi)}| =$ itm and 0 when $|ODB^{a,b}_{gr(\varphi)}| = 0$. Other functions sharing this property could be used as well.

There is nothing special with the equation we use when $|ODB^{a,b}_{gr(\varphi)}| \leq$ itm. The important thing is that arrives to 1 when $x =$ itm. Other equations can be used to model a more credulous or distrustful behaviour.

The *itm* value is domain dependent: it depends on the interaction frequency of the individuals in that society and also on the "quality" of those interactions. A plot of this function when $itm = 10$ is shown in figure 4.3.

The outcome deviation (noted as *Dv*) is the other factor that the ReGreT system takes into account to determine the reliability of a *direct trust* relationship. The greater the variability in the rating values the more volatile will the other agent be in the fulfill-

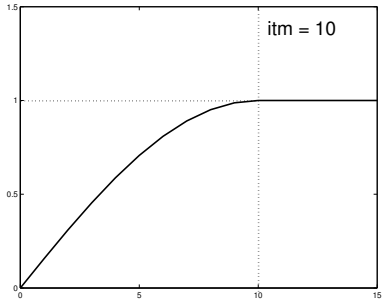

Figure 4.3: $No(ODB^{a,b}_{gr(\varphi)})$, $itm = 10$

ment of its agreements. To have a measure of this variability we consider the *impressions* of the outcomes that are used to calculate the *direct trust*.

We calculate the outcome reputation deviation as:

$$Dv(ODB^{a,b}_{gr(\varphi)}) = \sum_{o_i} \rho(t, t_i) \cdot |Imp(o_i, \varphi) - DT_{a \to b}(\varphi)|$$

Where $o_i \in ODB^{a,b}_{gr(\varphi)}$ and $Dv(ODB^{a,b}_{gr(\varphi)}) \in [0, 1]$. A deviation value near 1 indicates a high variability in the rating values (that is, a low credibility on the *direct trust* value from the outcome reputation deviation point of view) while a value close to 0 indicates a low variability (that is, a high credibility on the *direct trust* value). Note that we are calculating a kind of weighted mean deviation instead of a standard deviation.

Finally, we define the reliability of a *direct trust* relationship value (*DTRL*) as the product of functions *No* and (1-*Dv*).

$$DTRL_{a \to b}(\varphi) = No(ODB^{a,b}_{gr(\varphi)}) \cdot (1 - Dv(ODB^{a,b}_{gr(\varphi)}))$$

4.5 The reputation model

As we have seen, *direct trust* is used to build trust on the others based on direct experiences. The problem with direct experiences is that they are usually expensive to obtain in terms of time and cost (very often, the loss due to errors inherent to direct experimentation is not acceptable). In human societies people use the experience of others to overcome the lack of direct experiences. This aggregation of others' experience is the base of reputation. According to the concise oxford dictionary [Oxf, 2002], reputation is defined as "what is generally said or believed about a person's or thing's character or standing". The reputation model of the ReGreT system differentiates three types of reputation depending on the information source that is used to calculate them:

Witness Reputation. This reputation type is calculated using the information gathered from other agents that belong to the community. That information is the result of direct experiences that those agents had in the past with the target agent or to information that they gathered from other agents. We note this reputation type as: $R_{a \to b}(\varphi)$

Neighbourhood Reputation. It takes into account only the social environment of the target agent and the kind of relations the target agent has established with that environment. It is a reputation value based on prejudice because it is not based on the behaviour of the target agent itself but on the behaviour of those agents that have some kind of relation with it, together with the nature of that relation. The Spanish proverb "Dime con quien andas y te dire quien eres" (Tell me who you associate with and I will tell you who you are) illustrates the main idea behind this kind of reputation. It is noted as: $R_{a \xrightarrow{N} b}(\varphi)$

System Reputation. It is a default reputation based on objective features of the target agent (like the role it is playing in the society, the company that it belongs to and so on). It is not a reputation value based strictly on the target agent as an individual but on its membership to a certain group. We note this type of reputation as: $R_{a \xrightarrow{S} b}(\varphi)$

Usually the way to calculate system reputation and neighbourhood reputation is inherited from the group to which the agent belongs to as a kind of common knowledge. Therefore, both reputation types are representing the "way of thinking" of the group toward the rest of individuals and groups in the society.

Each one of these reputation types demands a different kind of knowledge of the agent society and the target agent. The *System reputation* is the easiest to calculate. We are assuming that the features used to calculate *system reputation* are known (or directly observable) by the other members of the society. The problem is that these features do not give enough information to compute a reputation on all imaginable aspects. For instance, if we know that somebody is a cook we can make suppositions about his/her skills in the kitchen or his/her knowledge about vegetables but we cannot say, for example, how good he/she is playing tennis. Also, the reliability of this type of reputation tends to be low because it doesn't take into account the peculiarities of the individual and its environment. This is the kind of reputation that an agent can use when it is a newcomer and there is an important lack of interaction with the other agents in the society. The *neighbourhood reputation* requires a deep knowledge of the environment of the target agent. This type of reputation can be useful in those situations where there is a good knowledge of the society and arrives a newcomer. Once the newcomer finds its place in the community it is possible to analyze its relations with known members of that community to extract some conclusions. Finally, *witness reputation* is based on word-of-mouth, with all the advantages and inconveniences associated to this kind of information.

Sociologically speaking, this division is far from complete. However, we consider that with these three types we maintain a good compromise between the complexity of the system and the requirements that an agent can satisfy in an e-commerce environment. In the following sections we explain in detail how each reputation type is calculated and how the ReGreT reputation model aggregates the information to obtain a single reputation value.

4.5.1 Witness reputation

Beliefs about trust can be (and usually are) shared among members of a society. The reputation that an agent builds on another agent based on the beliefs gathered from society members (witnesses) is what we call *witness reputation*. In an ideal world, with

only homogeneous and trusty agents, this information would be as relevant as direct experiences. However, in the kind of scenarios we are considering, it may happen that:

Information be wrong. Either because the other agents are trying to lie or because the information they own is not accurate, an agent has to be prepared to deal with wrong information.

Information be biased. The received information, although correct in essence, can be biased to favour the interests of the witness.

Agents hide information. An agent cannot assume that the information be complete.

Besides that, the information that comes from other agents can be correlated (what is called the *correlated evidence problem* [Pearl, 1988]). This happens when the opinions of different witnesses are based on the same event(s) or when there is a considerable amount of shared information that tends to unify the witnesses' way of "thinking". In both cases, the trust on the information shouldn't be as high as the number of similar opinions may suggest. As the event(s) that have generated the opinions for each agent may be hidden, the agent cannot identify directly which agents are correlated. Schillo et. al [Schillo et al., 2000] propose a method based on the analysis of "lying" as a stochastic process to implicitly reconstruct witness observations in order to alleviate this problem. We take a different approach based on the social relations between agents. Analysing these relations, an agent can obtain useful information to minimize the effects of the correlated evidence problem.

We assume that the information to be exchanged among agents is a tuple where the first element is the trust value on the target agent for a specific behavioural aspect from the point of view of the witness, and the second element is a value that reflects how confident the witness is about that trust value. We note the tuple as $\langle Trust_{w \to b}(\varphi), TrustRL_{w \to b}(\varphi) \rangle$, where w is the agent giving the information (the witness), b the target agent and φ the behavioural aspect considered. Each agent maintains a data base of received information. Similar to the outcomes data base, IDB^a is defined as the set of all information received by agent a and $IDB^{a,w}$ notes the subset of information received by agent a from agent w.

Identifying the witnesses

The first step to calculate a witness reputation is to identify the set of witnesses ($\mathbf{W}$) that will be taken into account by the agent to perform the calculation. The initial set of potential witnesses might be the set of all agents that have interacted with the target agent in the past. For instance, in an e-commerce environment, the initial set can be composed by all the agents that had had a trade relation with the target agent (it seems logical to think that the best witnesses about the commercial behaviour of the target agent are those agents that had a trade relation with it before). This set, however, can be very big and the information provided by its members probably suffer from the correlated evidence problem.

We take the stance that grouping agents with frequent interactions among them and considering each one of these groups as a single source of information minimizes the correlated evidence problem. Moreover, assuming that asking for information has a cost, it makes no sense to ask for the same thing to agents that we expect will give us

more or less the same answer. Grouping agents and asking for information to the most representative agent within each group reduces the number of queries to be done. A domain dependent sociogram is what we use to build these groups and to decide who is their most representative agent.

There are many heuristics that can be used to find groups and to select the best individual to ask. The heuristic used by the ReGreT witness reputation mechanism is based on the work by Hage and Harary [Hage and Harary, 1983]. Taking as the initial graph the subset of the selected sociogram over the agents that had interactions with the target agent, the heuristic is the following:

1. To identify the *components* of the graph. A *component* is defined as a maximally connected subgraph.

2. To find the set of *cut-points* (*CP*) for each component. A *cut-point* is a node whose removal would increase the number of components by dividing the sub-graph into two or more separate sub-graphs among which there are no connections. A cut-point can be seen from a sociological point of view as indicating some kind of *local centrality*. Cut-points are pivotal points of articulation between the agents that make up a component [Scott, 2000].

3. For each component that does not have cut-points, to choose as a representative for that component the node with the larger degree. If there is more than one node with the maximum degree, choose one randomly or, much better, you can use the *credibility* as a fitness measure for the selection (see section 4.5.1). This node is called a *central point*. Note that the degree can be regarded also as a measure of *local centrality* [Scott, 2000]. We refer to this set of nodes as *LCP*.

4. The set of selected nodes is the union of the set of *cut-points* and the set of *LCP*. That is, $\mathbf{W} = CP \cup LCP$.

Figure 4.4 shows an example of the application of the heuristic.

At this point, the agent has to evaluate if it is necessary to ask for renewed information to the members of the so calculated set of witnesses $\mathbf{W}$ or, on the contrary, the information available in the *IDB* is recent enough to be used.

We have proposed a simple heuristic strongly based on social network analysis. This heuristic doesn't take into account, for example, the availability of the witness or the economic cost associated to query that witness. In each domain it has to be considered if it is necessary to consider these aspects.

Who can I trust? The credibility model

Once the information is gathered from witnesses (or recovered from the data base of previous informations -*IDB*-), the agent obtains

$$\{\langle Trust_{w_i \to b}(\varphi),\, TrustRL_{w_i \to b}(\varphi)\rangle \mid w_i \in \mathbf{W}\}$$

where $\mathbf{W}$ is the subset of witnesses whom the agent has selected to be its sources of information. The next step is to aggregate these values to obtain a single value for the

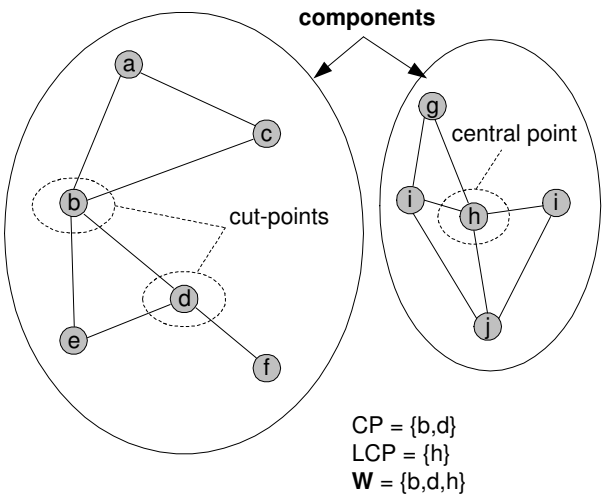

Figure 4.4: Witness selection within ReGreT.

witness Reputation. As we said before, however, it is possible that this information be wrong or biased. The agent has to be careful to give the right degree of reliability to each piece of information. The importance of each piece of information in the final reputation value will be proportional to the witness credibility.

Two different methods are used to evaluate the witness credibility.

The first method is based on the social structure among the witness, the target agent and the source agent. The idea is similar to that used to calculate the neighbourhood reputation (see section 4.5.2). We define $socialCr(a, w_i, b)$ as the credibility that agent a gives to w_i when w_i is giving information about b, taking only into account the social relations among a, w_i and b.

ReGreT uses fuzzy rules [Zadeh, 1975] to calculate how the structure of social relations influences the credibility on the information. The antecedent of each rule is the type and degree of a social relation (the edges in a sociogram) and the consequent is the credibility of the witness from the point of view of that social relation. For example:

IF $coop(w_i, b)$ is high
THEN $socialCr(a, w_i, b)$ is very_low

that is, if the level of cooperation between w_i and b is high then the credibility that the information coming from w_i related to b has, from the point of view of a, is very low. The heuristic behind this rule is that a cooperative relation implies some degree of complicity between the agents that share this relation so the information coming from one about the other is probably biased.

Which relations are relevant to calculate the credibility depends on the meaning that each relation type has in the specific agent community. In a *SuppWorld* scenario, for instance, a trade relation cannot cast any light on the credibility of the information

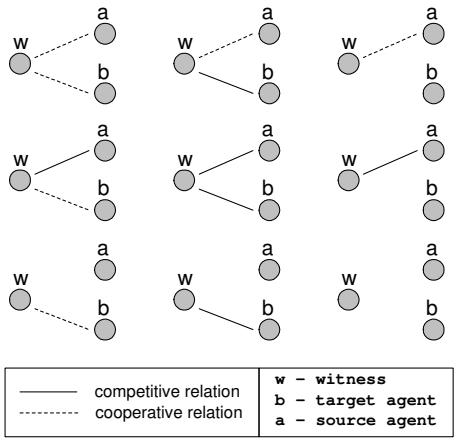

Figure 4.5: Relevant social structures in a SuppWorld scenario to evaluate credibility.

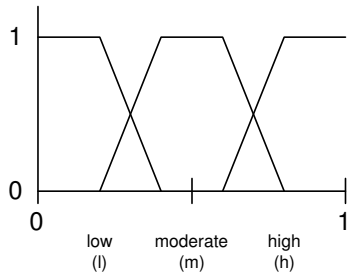

Figure 4.6: Intensity of a social relation.

coming from the agents involved in that relation (always from the point of view of social analysis). In other scenarios, however, this could be the other way around.

Following with the SuppWorld scenario, from the set of social relations only the cooperative relation (*coop*) and the competitive (*comp*) relation are relevant to calculate a measure of credibility. Hence, together with the "no relation" (no_rel) possibility there are 9 social structures to be considered as shown in Figure 4.5.

Figure 4.6 shows the fuzzy sets —that give the meaning of the intensity labels used on the arcs of the sociogram— for the values $coop(w_i, a)$, $coop(w_i, b)$, $comp(w_i, a)$, and $comp(w_i, b)$, and figure 4.7 shows the fuzzy sets for the variable $socialCr\,(a, w_i, b)$. The variable no_rel is boolean. Table 4.1 shows a possible set of fuzzy rules. Note that a great percentage of the rules tend to be "pessimistic". This is because in those cases where it is not clear that the behaviour is going to be good, we think it is preferable to be cautious. At this moment the kind of influence of each social structure is hand-coded and based on human common sense. An improvement would be the use of a rule learning mechanism to automate the process.

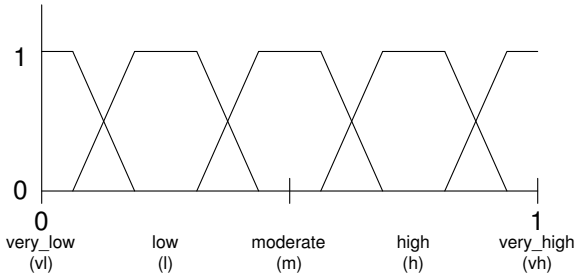

Figure 4.7: Fuzzy sets for the variable $socialCr(a, w_i, b)$.

IF	$coop(w_i, a)$ is l	THEN	$socialCr(a, w_i, b)$ is h
IF	$coop(w_i, a)$ is m	THEN	$socialCr(a, w_i, b)$ is vh
IF	$coop(w_i, a)$ is h	THEN	$socialCr(a, w_i, b)$ is vh
IF	$comp(w_i, a)$ is l	THEN	$socialCr(a, w_i, b)$ is m
IF	$comp(w_i, a)$ is m	THEN	$socialCr(a, w_i, b)$ is l
IF	$comp(w_i, a)$ is h	THEN	$socialCr(a, w_i, b)$ is vl
IF	$coop(w_i, b)$ is l	THEN	$socialCr(a, w_i, b)$ is m
IF	$coop(w_i, b)$ is m	THEN	$socialCr(a, w_i, b)$ is l
IF	$coop(w_i, b)$ is h	THEN	$socialCr(a, w_i, b)$ is vl
IF	$comp(w_i, b)$ is l	THEN	$socialCr(a, w_i, b)$ is m
IF	$comp(w_i, b)$ is m	THEN	$socialCr(a, w_i, b)$ is l
IF	$comp(w_i, b)$ is h	THEN	$socialCr(a, w_i, b)$ is vl
IF	$no\_rel(w_i, b)$	AND	$no\_rel(w_i, a)$
		THEN	$socialCr(a, w_i, b)$ is h

Table 4.1: Social credibility fuzzy rules.

The second method used in the ReGreT system to calculate the credibility of a witness is to evaluate the accuracy of previous pieces of information sent by that witness to the agent. The agent is using the *direct trust* value (see section 4.4) to measure the truthfulness of the information received from witnesses. For example, an agent a receives information from witness w about agent b saying agent b offers good quality products. Later on, after interacting with agent b, agent a realizes that the products that agent b is selling are horrible. This will be reflected in the value of the *direct_trust* associated to the aspect *offers_good_quality* $\langle DT_{a\to b}(\textit{offers\_good\_quality}), DTRL_{a\to b}(\textit{offers\_good\_quality})\rangle$. If the *direct_trust* value is low (near -1) it means agent b is offering bad products and therefore that agent w was giving wrong information.

Summarizing, what an agent a is using to evaluate the accuracy of a witness w are pairs of tuples of the form:

$$\langle Trust_{w\to b}(\varphi), TrustRL_{w\to b}(\varphi)\rangle$$
$$\langle DT_{a\to b}(\varphi), DTRL_{a\to b}(\varphi)\rangle$$

with $b \in B$ and $\varphi \in \Phi$, where B is the set of agents in that society and Φ the set of behavioural aspects.

One important property that has to be remarked about these tuples is that they are not static. They change through time either because the agent collects more direct experiences that modify the perspective it has on the target agent (giving or not more credibility to the witness) or because the agent obtains new information from the witness that overwrites the previous one. Only the most recent information referred to a specific target agent and behavioural aspect from a given witness is stored in the information data base (*IDB*). Giving the witnesses the opportunity to rectify previous information we are allowing them to correct previous mistakes.

By comparing the trust value assigned by the witness with its own perception of the target agent (represented by the *direct trust* value) the agent obtains the degree of truth of that piece of information. However, there is an important aspect we have not considered up to now. When the trust values are very different but the reliability assigned by the witness is very high and the reliability of the *direct trust* is very low, the agent should decrease the credibility of the witness when it is almost sure that the *direct trust* value the agent has calculated is wrong due to lack of knowledge? What happens if it is the other way around? Is it sensible to decrease the trustworthiness of the witness when the witness itself was giving advice about the weakness of the information by means of the reliability value? Clearly, the method used to evaluate the accuracy of a piece of information has to take much into account the reliability values associated to the trust values in order to decide when the accuracy measure is relevant or not.

There are three main situations the model has to consider:

- $DTRL \approx 0$. The agent does not have enough direct knowledge to judge the truthfulness of what the witness is saying.
- $TrustRL \approx 0$. The witness recognizes the weakness of the given information. Therefore that information cannot be used to judge the credibility of the witness.
- $DTRL \approx 1$, $TrustRL \approx 1$. The witness is very confident about the information and the agent has enough direct knowledge to judge the truthfulness of that information (and therefore the credibility of the witness).

This can be easily modelled using the product between *TrustRL* and *DTRL* as a factor of relevance for the comparison. Given a piece of information $I = \langle Trust_{w \to b}(\varphi), TrustRL_{w \to b}(\varphi) \rangle \in IDB^{a,w}$, we define the relevance of that information as:

$$\sigma_I = TrustRL_{w \to b}(\varphi) \cdot DTRL_{a \to b}(\varphi)$$

The formula used in the ReGreT system to evaluate the credibility of a witness considering the accuracy of previous information received from that witness is:

$$infoCr(a, w) = \frac{\sum_{I \in IDB^{a,w}_{\sigma > 0.5}} \sigma_I \cdot Ap_0(Trust_{w \to b}(\varphi) - DT_{a \to b}(\varphi))}{\sum_{I \in IDB^{a,w}_{\sigma > 0.5}} \sigma_I}$$

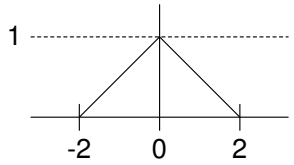

Figure 4.8: Ap_0 function.

where $IDB^{a,w}_{\sigma>0.5}$ is defined as $\{I \in IDB^{a,w} : \sigma_I > 0.5\}$. Imposing the restriction of using only those pieces of information with a relevance greater than 0.5 we ensure a minimum quality on the result. The function Ap_0 is depicted in fig 4.8. If the difference $(Trust_{w\to b}(\varphi) - DT_{a\to b}(\varphi))$ is near 0 it means the witness coincides with the agent (and therefore we have a value for the credibility near 1), on the contrary if the difference shows a value near 2 or -2, it means the witness information is different to what the agent has experienced by itself. The conclusion is that the witness is lying and we obtain value for the credibility of that witness (always associated to that specific piece of information) near 0.

Similar to the case of the *direct trust*, the ReGreT system calculates a measure of reliability for the credibility value *infoCr*. Again, we use the number of values considered for the calculation and the variability of those values as a measure of that reliability. The formula to calculate the reliability of a given *infoCr* value is:

$$infoCrRL(a,w) = Ni(IDB^{a,w}_{\sigma>0.5}) \cdot (1 - Dv(IDB^{a,w}_{\sigma>0.5}))$$

where

$$Ni(IDB^{a,w}_{\sigma>0.5}) = \begin{cases} \sin\left(\frac{\pi \cdot |IDB^{a,w}_{\sigma>0.5}|}{2 \cdot itm}\right) & |IDB^{a,w}_{\sigma>0.5}| \leq \text{itm} \\ 1 & \text{otherwise} \end{cases}$$

$$Dv(IDB^{a,w}_{\sigma>0.5}) = \frac{\sum_{I \in IDB^{a,w}_{\sigma>0.5}} (\sigma_I \cdot |\mathcal{A}|)}{\sum_{I \in IDB^{a,w}_{\sigma>0.5}} \sigma_I}$$

with $\mathcal{A} = Ap_0(Trust_{w\to b}(\varphi) - DT_{a\to b}(\varphi)) - infoCr(a,w)$.

We consider that the credibility calculated considering the accuracy of previous pieces of information (*infoCr*) is more reliable than the credibility based on social relations (*socialCr*). While the analysis of social relations is based on expected behaviours, the analysis of previous information is based on particular facts from the witness the agent wants to evaluate. However, in those situations where there is not enough information to calculate a reliable *infoCr* value, the analysis of social relations can be a good solution. Usually, social relations are easier to obtain than the necessary information to calculate a reliable *infoCr* value. To define the credibility that a witness w_i deserves to an agent a when it is giving information about an agent b we have to differentiate several possibilities:

- Both values ($infoCr$ and $socialCr$) can be calculated. The agent uses the *infoCr* value if it is reliable, if not, it uses the credibility based on social relations. The formula for this situation is:

$$\begin{aligned} witnessCr(a, w_i, b) &= infoCrRL(a, w_i) \cdot infoCr(a, w_i) + \\ &\quad (1 - infoCrRL(a, w_i)) \cdot socialCr(a, w_i, b) \end{aligned}$$

- The $socialCr$ value is not available (the agent does not have enough social information to calculate it). In this situation we have to differentiate two cases:

$$\begin{aligned} \text{If } (infoCrRL > 0.5) &\qquad witnessCr(a, w_i, b) = infoCr(a, w_i) \\ \text{Otherwise} &\qquad witnessCr(a, w_i, b) = 0.5 \end{aligned}$$

- The $infoCr$ value is not available (the agent does not have direct experiences to evaluate if the information from the witness is reliable or not). Again there are two possibilities:

$$\begin{aligned} \text{If } (socialCr \text{ is available}) &\qquad witnessCr(a, w_i, b) = socialCr(a, w_i) \\ \text{Otherwise} &\qquad witnessCr(a, w_i, b) = 0.5 \end{aligned}$$

The default value of 0.5 used when there is not enough information to judge the credibility of a witness depends on how credulous the agent is.

Witness reputation

Now we have all the elements to calculate a *witness reputation* and its associated reliability value considering that the information coming from the witnesses can be wrong or biased. The formulae in the ReGreT system to calculate these values are:

$$R_{a \overset{W}{\rightarrow} b}(\varphi) = \sum_{w_i \in \mathbf{W}} \omega^{w_i b} \cdot Trust_{w_i \rightarrow b}(\varphi)$$

$$RL_{a \overset{W}{\rightarrow} b}(\varphi) = \sum_{w_i \in \mathbf{W}} \omega^{w_i b} \cdot \min(witnessCr(a, w_i, b), TrustRL_{w_i \rightarrow b}(\varphi))$$

where $\omega^{w_i b} = \frac{witnessCr(a, w_i, b)}{\sum_{w_j \in \mathbf{W}} witnessCr(a, w_j, b)}$

These formulae require some explanations. To calculate a *witness reputation* the agent uses the normalized credibility of each witness to weight its opinion in the final value. For the calculation of the reliability, we want that each individual contributes in the same proportion that it has contributed for the calculation of the reputation value. Therefore, the agent uses in the reliability formula the same weights that are used in the reputation formula.

To calculate the reliability of a witness opinion, the agent uses the minimum between the witness credibility and the reliability value that the witness itself provides. If the witness is a trusty agent, the agent can use the reliability value the witness has proposed. If not, the agent will use the credibility of the witness as a measure for the reliability of the information.

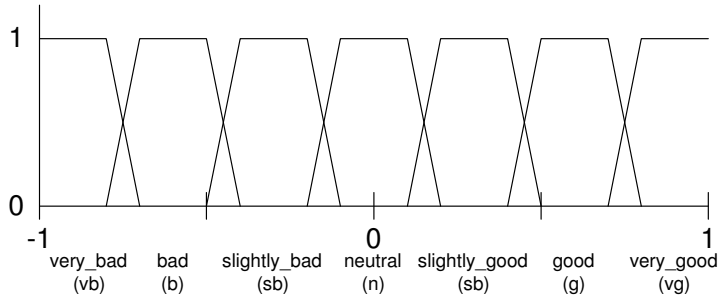

Figure 4.9: Fuzzy sets for variables $DT_{a \to n_i}$ and $Rep_{a \overset{n_i}{\to} b}$.

4.5.2 Neighbourhood reputation

The trust on the agents that are in the neighbourhood of the target agent and their relation with it are the elements used to calculate what we call the *Neighbourhood Reputation*. Neighbourhood in a MAS is not related with the physical location of the agents but with the links created through interaction. The main idea is that the behaviour of these neighbours and the kind of relation they have with the target agent can give some clues about the behaviour of the target agent. We note the set of neighbours of agent b as $\mathbf{N}_b = \{n_1, n_2, \cdots, n_n\}$.

To calculate a *Neighbourhood Reputation* the ReGreT system uses fuzzy rules. The antecedents of these rules are one or several *direct trusts* associated to different behavioural aspects and the relation between the target agent and the neighbour. The consequent is the value for a concrete reputation (that can be associated to the same behavioural aspect of the trust values or not).

The application of these rules generates a set of *individual neighbourhood reputations* noted as $R_{a \overset{n_i}{\to} b}(\varphi)$. For instance, using again the SuppWorld scenario, an example could be:

IF $DT_{a \to n_i}(offers\_good\_quality)$ is X AND $coop(b, n_i) \geqslant$ low
THEN $R_{a \overset{n_i}{\to} b}(offers\_good\_quality)$ is X
IF $DTRL_{a \to n_i}(offers\_good\_quality)$ is X' AND $coop(b, n_i)$ is Y'
THEN $RL_{a \overset{n_i}{\to} b}(offers\_good\_quality)$ is T(X', Y')

In other words, we are saying that if the neighbour of the target agent is offering good quality products and there is a relation of cooperation between the target and this neighbour, then the target is also assumed to offer good quality products. Here, a neighbour of agent b is an agent that has a *coop* relation with it. The fuzzy sets for variables $DT_{a \to n_i}$ and $R_{a \overset{n_i}{\to} b}$ are shown in figure 4.9 and the fuzzy sets for variable $RL_{a \overset{n_i}{\to} b}$ are shown in figure 4.10.

Finally table 4.2 shows a possible set of values for function T.

As we have said, instead of relying on the performed actions of the target agent, *Neighbourhood reputation* is using prejudice as a mechanism for evaluation. In human societies the word "prejudice" refers to a negative or hostile attitude toward another

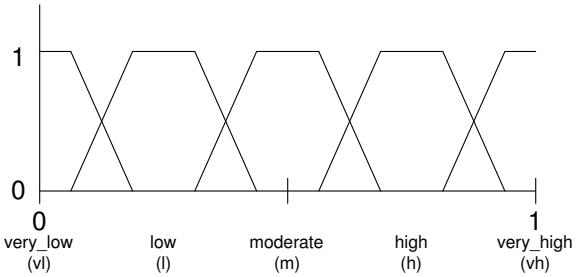

Figure 4.10: Fuzzy sets for variable $RL_{a \xrightarrow{n_i} b}$

X' Y'	l	m	h
vl	vl	vl	vl
l	vl	vl	l
m	vl	l	m
h	l	m	h
vh	m	h	vh

Table 4.2: Function T used in reliability rules.

social group, usually racially defined. However we are not talking about human societies where prejudice is without any doubt blameworthy. We are talking about virtual environments populated by software agents. We think that the use of prejudice in the context of agents has a positive aspect. If an agent knows the others are judging it in part because of its partners, it will be careful to choose the right partners and avoid cheaters that perhaps at the beginning can offer better deals but at the end will deteriorate its reputation in front of the community. Moreover, the modular design of the ReGreT reputation model allows to cancel the influence of one type of reputation (in this case the *Neighbourhood reputation*) if it is not useful or convenient in a given environment.

The general formulae we use to calculate a *neighbourhood reputation* and its reliability are similar to those used to calculate a *witness reputation*:

$$R_{a \xrightarrow{N} b}(\varphi) = \sum_{n_i \in N_b} \omega^{n_i b} \cdot R_{a \xrightarrow{n_i} b}(\varphi)$$

$$RL_{a \xrightarrow{N} b}(\varphi) = \sum_{n_i \in N_b} \omega^{n_i b} \cdot RL_{a \xrightarrow{n_i} b}(\varphi)$$

where $\omega^{n_i b} = \frac{RL_{a \xrightarrow{n_i} b}(\varphi)}{\sum_{n_j \in N_b} RL_{a \xrightarrow{n_j} b}(\varphi)}$

In this case we are using the reliability of each *neighbourhood reputation* value to weight the contribution to the final result, both for the reputation and the reliability.

4.5.3 System reputation

The idea behind *System reputations* is to use the common knowledge about *social groups* and the role that the agent is playing in the society as a mechanism to assign

default reputations to the agents. We assume that the members of these groups have one or several *observable* features that unambiguously identify their membership. The idea behind *system reputation* is similar to the idea behind *neighbourhood reputation*. As we have seen, *Neighbourhood reputation* focus on reduced groups of agents where the links between their members can not always be easily recognized by the agents that do not belong to the group. On the contrary, the groups considered by *system reputation* are usually mid to big sized and their members can be easily identified. We assume that the role that an agent is playing and the group (or groups) it belongs to is something "visible" and unambiguous for the other agents in that society.

Each time an agent performs an action we consider that it is playing a single role. An agent can play the role of buyer and seller but when it is selling a product only the role of seller is relevant. Although we can think up some situations where an agent can play two or more different roles at a time, we consider that there is always a predominant role and the others can be disregarded.

The knowledge necessary to calculate a *system reputation* is usually inherited from the group or groups to which the agent belongs to. Each group provides knowledge about different aspects. We share the stance that groups influence the point of view of their members [Karlins and I.Abelson, 1970].

System reputations are calculated using a table for each social group where the rows are the roles the agent can play for that group, and the columns the behavioural aspects.

Table 4.3 shows an example of *system reputations* for agents that belong to company B from the point of view of an agent of company A. As you notice, in this case the opinion of company A toward agents in company B is not very good.

	offers_good_prices	***offers_good_quality***	***delivers_quickly***	***pays_on_time***
seller	-0.6	-0.8	-0.6	-
buyer	-	-	-	-0.6

Table 4.3: Example of *system reputations.*

Using a similar table we would define the reliability for these reputations.

System reputations are noted as $R_{a \overset{S}{\rightarrow} b}(\varphi)$ and its reliability as $RL_{a \overset{S}{\rightarrow} b}(\varphi)$. Hence, for example, using the table defined above, we have that $R_{a \overset{S}{\rightarrow} b}(pays\_on\_time) = -0.6$ where b is a buyer that belongs to company B.

The degree of influence that the group or groups to which the agent belongs to have on it, will fix the reliability assigned to *system reputation*.

4.5.4 Combining reputation types

In the previous section we have gone through the three different reputation types considered in the ReGreT reputation model. To these reputation types we have to add a fourth one, the reputation assigned to a third party agent when there is no information at all: the *default* reputation. This reputation is noted as $R_{a \overset{D}{\rightarrow} b}(\varphi)$. Usually this will be a fixed value for all b and φ values, however we give the possibility to assign different default reputation values depending on the behavioural aspect (for example, in certain

situations the agent could be more trusting). Anyway, what is important is that the default reputation is always available. Similarly to other reputation types, there is also a reliability value associated to the default reputation noted as $RL_{a \overset{D}{\rightarrow} b}(\varphi)$.

In this section we will show how these reputations are combined to obtain a single reputation value. As we have seen, each reputation type has different characteristics and there are a lot of heuristics that can be used to aggregate the four reputation values to obtain a single and representative reputation value. The heuristic we propose here is based on the default and calculated reliability assigned to each type.

Assuming we have enough information to calculate all the reputation types, we have the stance that *witness reputation* is the first type that should be considered followed by the *neighbourhood reputation*, *system reputation* and finally the *default reputation*. This ranking, however, has to be subordinated to the calculated reliability for each type.

Given that, we define the reputation that an agent a assigns to an agent b associated to certain behavioural aspect φ as:

$$R_{a \rightarrow b}(\varphi) = \sum_{i \in \{W,N,S,D\}} \xi_i \cdot R_{a \overset{i}{\rightarrow} b}(\varphi)$$

and similarly for reliability:

$$RL_{a \rightarrow b}(\varphi) = \sum_{i \in \{W,N,S,D\}} \xi_i \cdot RL_{a \overset{i}{\rightarrow} b}(\varphi)$$

Following the ranking we have established before, the factors $\{\xi_W, \xi_N, \xi_S, \xi_D\}$ we use in the general formula are:

$$\begin{aligned}
\xi_W &= RL_{a \overset{W}{\rightarrow} b}(\varphi) \\
\xi_N &= RL_{a \overset{N}{\rightarrow} b}(\varphi) \cdot (1 - \xi_W) \\
\xi_S &= RL_{a \overset{S}{\rightarrow} b}(\varphi) \cdot (1 - \xi_W - \xi_N) \\
\xi_D &= 1 - \xi_W - \xi_N - \xi_S
\end{aligned}$$

That is, we want the agent to give more relevance to the *witness reputation* in detriment of the others. If the *witness reputation* has a low degree of reliability (for instance because the witnesses are not reliable) then the agent will try to use the *neighbourhood reputation*. If the agent has a poor knowledge of the social relationships and as result of that the reliability of the *neighbourhood reputation* is low, it will try to use the *system reputation*. Finally it will use the *default reputation*.

4.6 Putting all together: the trust model

As showed in figure 4.1 the ReGreT system considers two elements to calculate the trust on an agent: the reputation of that agent and the *direct trust* (that is, the result of direct experiences).

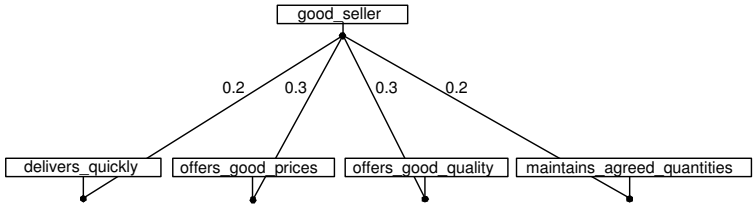

Figure 4.11: Ontological structure for a buyer in the *SuppWorld* scenario.

As we have argued, *direct trust* is a more reliable source of information than reputation. Using the same approach that for the reputation calculation we define the trust that an agent b deserves to an agent a on certain behavioural aspect φ as:

$$\begin{aligned} Trust_{a\to b}(\varphi) &= DTRL_{a\to b}(\varphi)\cdot DT_{a\to b}(\varphi) + \\ &\quad (1 - DTRL_{a\to b}(\varphi))\cdot R_{a\to b}(\varphi) \end{aligned}$$

$$\begin{aligned} TrustRL_{a\to b}(\varphi) &= DTRL_{a\to b}(\varphi)\cdot DTRL_{a\to b}(\varphi) + \\ &\quad (1 - DTRL_{a\to b}(\varphi))\cdot RL_{a\to b}(\varphi) \end{aligned}$$

If the agent has a reliable *direct trust* value, it will use that as a measure of trust. If that value is not so reliable then it will use reputation.

4.7 Ontological dimension

Up to now we have shown how to calculate reputation and trust values linked to behavioural aspects that refer to a single issue of a contract. With the ontological dimension we add the possibility of combining these reputations and trust values associated to simple behaviours to calculate the reputation and trust of more complex behaviours.

To represent the *ontological* dimension we use graph structures. Figure 4.11 shows an example of a simple ontology structure for a buyer in the *SuppWorld* scenario.

In this case, being a good seller implies delivering products quickly, offering good products, offering good quality and maintain agreed quantities. The buyer gives more relevance to the quality and price of the products to decide if a seller is a good seller or not.

To calculate a given trust taking into account the *ontological dimension*, an agent has to calculate the value of each of the related aspects that, in turn, can be the node of another subgraph with other associated aspects. The trust values for those nodes that are related with an atomic aspect of the behaviour (in the example: *deliver_quickly*, *offer_good_prices*, *offer_good_quality* and *maintain_agreed_quantities*), are calculated using the methods we have presented in the previous sections. Note that an ontology structure can be applied to different parts of the system. The agent can use the ontology either to calculate a reputation value or to calculate a trust value.

The trust over an internal node ψ is computed as follows:

$$Trust_{a\to b}(\psi) = \sum_{\varphi \in children(\psi)} \omega_{\psi\varphi} \cdot Trust_{a\to b}(\varphi)$$

$$TrustRL_{a\to b}(\psi) = \sum_{\varphi \in children(\psi)} \omega_{\psi\varphi} \cdot TrustRL_{a\to b}(\varphi)$$

For instance, using the ontological structure in figure 4.11 we can calculate the trust on b as a good seller from a's perspective using the formula:

$$\begin{aligned} Trust_{a\to b}(good\_seller) &= 0.3 \cdot Trust_{a\to b}(deliver\_quickly) + \\ &\quad 0.4 \cdot Trust_{a\to b}(offer\_good\_prices) + \\ &\quad 0.4 \cdot Trust_{a\to b}(offer\_good\_quality) + \\ &\quad 0.3 \cdot Trust_{a\to b}(maintain\_agreed\_quantities) \end{aligned}$$

$$\begin{aligned} TrustRL_{a\to b}(good\_seller) &= 0.3 \cdot TrustRL_{a\to b}(deliver\_quickly) + \\ &\quad 0.4 \cdot TrustRL_{a\to b}(offer\_good\_prices) + \\ &\quad 0.4 \cdot TrustRL_{a\to b}(offer\_good\_quality) + \\ &\quad 0.3 \cdot TrustRL_{a\to b}(maintain\_agreed\_quantities) \end{aligned}$$

The same ontological structure could be used to calculate the reputation of being a *good_seller*.

Note that the importance ($\omega_{\psi\varphi}$) of each aspect is agent dependent and not necessarily static. The agent can change these values according to its mental state.

Chapter 5

Infrastructure: the agent model

In previous chapters we have presented the ReGreT system. Although, as we have seen, the system can be used by simple agents in relatively simple environments, it is only in complex societies where the system can show its potential. The agents that populate this kind of societies must have a deliberative component with a certain degree of complexity. Therefore, the specification and implementation of these agents is not an easy task. Chapters 5 and 6 make a proposal in that direction.

5.1 Introduction

Agent-based computing is fast emerging as a new paradigm for engineering complex, distributed systems [Jennings, 1999, Wooldridge, 1997]. An important aspect of this trend is the use of agent architectures as a means of delivering agent-based functionality (cf. work on agent programming languages [Meyer, 1998, Thomas, 1995, Weerasooriya et al., 1995]). In this context, an architecture can be viewed as a separation of concerns—it identifies the main functions that ultimately give rise to the agent's behaviour and defines the interdependencies that exist between them. As agent architectures become more widely used, there is an increasing demand for unambiguous specifications of them and there is a greater need to verify their implementations. To this end, a range of techniques have been used to formally specify agent architectures (eg Concurrent MetateM [Fisher, 1998, Wooldridge, 1996], DESIRE [Brazier et al., 1995, Treur, 1991] and Z [d'Inverno et al., 1998]). However, these techniques typically fall short in at least one of the following ways: (i) they enforce a particular view of architecture upon the specification; (ii) they offer no explicit structures for modelling the components of an architecture or the relationships between them; (iii) they leave a gap between the specification of an architecture and its implementation.

To rectify these shortcomings, Parsons et al. proposed [Parsons et al., 1998] the use of *multi-context systems* [Giunchiglia and Serafini, 1994] as a means of specifying and implementing agent architectures. Multi-context systems provide an overarching framework that allows distinct theoretical components to be defined and interrelated.

Such systems consist of a set of contexts, each of which can informally be considered to be a logic and a set of formulae written in that logic, and a set of bridge rules for transferring information between contexts. Thus, different contexts can be used to represent different components of the architecture and the interactions between these components can be specified by means of the bridge rules between the contexts. We believe multi-context systems are well suited to specifying and modelling agent architectures for two main types of reason: (i) from a *software engineering perspective* they support modular decomposition and encapsulation; and (ii) from a *logical modelling perspective* they provide an efficient means of specifying and executing complex logics. Each of these broad areas will now be dealt with in turn.

Let us first consider the advantages from a software engineering perspective. Firstly, multi-context systems support the development of modular architectures. Each architectural component—be it a functional component (responsible for assessing the agent's current situation, say) or a data structure component (the agent's beliefs, say)—can be represented as a separate context. The links between the components can then be made explicit by writing bridge rules to link the contexts. This ability to directly support component decomposition and component interaction offers a clean route from the high level specification of the architecture through to its detailed design. Moreover, this basic philosophy can be applied no matter how the architectural components are decomposed or how many architectural components exist. Secondly, since multi-context systems encapsulate architectural components and enable flexible interrelationships to be specified, they are ideally suited to supporting re-use (both of designs and implementations). Thus, contexts that represent particular aspects of the architecture can be packaged as software components (in the component-ware sense [Szyperski, 1998]) or they can be used as the basis for specialization of new contexts (inheritance in the object-oriented sense [Booch, 1994]).

Moving onto the logical modelling perspective, there are four main advantages of adopting a multi-context approach. The first is an extension of the software engineering advantages which specifically applies to logical systems. By breaking the logical description of an agent into a set of contexts, each of which holds a set of related formulae, we effectively get a form of many-sorted logic (all the formulae in one context are a single sort) with the concomitant advantages of scalability and efficiency. The second advantage follows on from this. Using multi-context systems makes it possible to build agents which use several different logics in a way that keeps the logics neatly separated (all the formulae in one logic are gathered together in one context). This either makes it possible to increase the representational power of logical agents (compared with those which use a single logic) or simplify agents conceptually (compared with those which use several logics in one global context). This latter advantage is illustrated in [Parsons et al., 1998] where multi-context systems are used to simplify the construction of a belief/desire/intention (BDI) agent.

Both of the above advantages apply to any logical agent built using multi-context systems. The remaining two advantages apply to specific types of logical agents—those which reason about their mental attitudes and those of other agents. The first is that multi-context systems make it possible [Giunchiglia and Serafini, 1994] to build agents which reason in a way which conforms to the use of modal logics like KD45

(the standard modal logic for handling belief) but which obviates the difficulties usually inherent in theorem proving in such logics. Thus the use of multi-context systems makes it easy to directly execute agent specifications where those specifications deal with modal notions. Again this is illustrated in [Parsons et al., 1998]. The final advantage is related to this. Agents which reason about beliefs are often confronted with the problem of modelling the beliefs of other agents, and this can be hard, especially when those other agents reason about beliefs in a different way (because, for instance, they use a different logic). Multi-context systems provide a neat solution to this problem [Benerecetti et al., 1997, Cimatti and Serafini, 1995].

When the software engineering and the logical modelling perspectives are combined, it can be seen that the multi-context approach offers a clear path from specification through to implementation. By providing a clear set of mappings from concept to design, and from design to implementation, the multi-context approach offers a way of tackling the gap that currently exists between the theory and the practice of agent-based systems. To some extent the advantages of multi-context systems were explored in [Parsons et al., 1998]. Here, we extend the former by further refining the approach, extending the representation and providing additional support for building complex agents. In particular we introduce three new ideas. The first is that of grouping contexts together into modules, giving another level of abstraction in defining agent architectures. The second is the idea of bridge rules which delete formulae from certain contexts (as opposed to just introducing them), an idea which allows the modelling of consumable resources. The third idea is that of introducing a time-delay into the execution of a bridge rule in order to allow inter-context synchronisation.

The remainder of this chapter is structured in the following manner. Section 5.2 introduces the ideas of multi-context systems on which our approach is founded. Section 5.3 explains how we have extended the use of multi-context systems to better handle systems of high complexity. Section 5.4 then illustrates our approach using a specific agent architecture and a specific exemplar scenario, and Section 5.5 extends this example to include inter agent communication. Finally, section 5.6 compares our approach to other proposals in more or less the same vein.

5.2 Multi-context agents

As discussed above, we believe that the use of multi-context systems offers a number of advantages when engineering agent architectures. However, multi-context systems are not a panacea. We believe that they are most appropriate when building agents which are logic-based and are therefore largely deliberative. Whether such agents are the best solution depends on the task the agent is to perform. See [Wooldridge and Jennings, 1995] for a discussion of the relative merits of logic-based and non logic-based approaches to specifying and building agent architectures.

5.2.1 The basic model

Using a multi-context approach, an agent architecture consists of four basic types of component. These components were first identified in the context of building theorem

provers for modal logic [Giunchiglia and Serafini, 1994], before being identified as a methodology for constructing agent architectures [Noriega and Sierra, 1996] where full detail of the components can be found. In brief, the components are the following:

- *Units*: Structural entities representing the main components of the architecture.
- *Logics*: Declarative languages, each with a set of axioms and a number of rules of inference. Each unit has a single logic associated with it.
- *Theories*: Sets of formulae written in the logic associated with a unit.
- *Bridge rules*: Rules of inference which relate formulae in different units.

Units represent the various components of the architecture. They contain the bulk of an agent's problem solving knowledge, and this knowledge is encoded in the specific theory that the unit encapsulates. In general, the nature of the units will vary between architectures. For example, a BDI agent may have units which represent theories of beliefs, desires and intentions (as in [Parsons et al., 1998]), whereas an architecture based on a functional separation of concerns may have units which encode theories of cooperation, situation assessment and plan execution. In either case, each unit has a suitable logic associated with it. Thus the belief unit of a BDI agent has a logic of belief associated with it, and the intention unit has a logic of intention. The logic associated with each unit provides the language in which the information in that unit is encoded, and the bridge rules provide the mechanism by which information is transferred between units.

Bridge rules can be understood as rules of inference with premises and conclusions in different units. For instance:

$$\frac{u_1 : \psi, u_2 : \varphi}{u_3 : \theta}$$

means that formula θ may be deduced in unit u_3 if formulae ψ and φ are deduced in units u_1 and u_2 respectively.

When used as a means of specifying agent architectures [Noriega and Sierra, 1996, Parsons et al., 1998], all the elements of the model, both units and bridge rules, are taken to work concurrently. In practice this means that the execution of each unit is a non-terminating, deductive process. The bridge rules continuously examine the theories of the units that appear in their premises for new sets of formulae that match them. This means that all the components of the architecture are always ready to react to any change (external or internal) and that there are no central control elements.

5.2.2 The extended model

The model as outlined above is that introduced in [Noriega and Sierra, 1996] and used in [Parsons et al., 1998]. However, this model has proved deficient in a couple of ways, both connected to the dynamics of reasoning. In particular we have found it useful to extend the basic idea of multi-context systems by associating two control elements with the bridge rules: *consumption* and *time-outs*. A consuming condition means that the bridge rule removes the formula from the theory which contains the premise (remember

that a theory is considered to be a set of formulae). Thus in bridge rules with consuming conditions, formulae "move" between units. To distinguish between a consuming condition and a non-consuming condition, we use the notation $u_i > \psi$ for consuming and $u_i : \psi$ for non-consuming conditions. Thus:

$$\frac{u_1 > \psi, u_2 : \varphi}{u_3 : \theta}$$

means that when the bridge rule is executed, ψ is removed from u_1 but φ is not removed from u_2.

Consuming conditions increase expressiveness in the communication between units. With this facility, we can model the movement of a formula from one theory to another (from one unit to another), changes in the theory of one unit that cause the removal of a formula from another one, and so on. This mechanism also makes it possible to model the concept of state since having a concrete formula in one unit or another might represent a different agent state. For example, later in the chapter we use the presence of a formula in a particular unit to indicate the availability of a resource.

A time-out in a bridge rule means there is a delay between the instant in time at which the conditions of the bridge rule are satisfied and the effective activation of the rule. A time-out is denoted by a label on the right of the rule; for instance:

$$\frac{u_1 : \psi}{u_2 : \varphi}[t]$$

means that t units of time after the theory in unit u_1 gets formula ψ, the theory in unit u_2 will be extended by formula φ. If during this time period formula ψ is removed from the theory in unit u_1, this rule will not be applied. In a similar way to consuming conditions, time-outs increase expressiveness in the communication between units. This is important when actions performed by bridge rules need to be retracted if a specific event does not happen after a given period of time. In particular, it enables us to represent situations where silence during a period of time may mean failure (in this case the bridge rules can then be used to re-establish a previous state).

Both of these extensions to the standard multi-context system incur a cost. This is that including them in the model means that the model departs somewhat from first order predicate calculus, and so does not have a fully-defined semantics. One possibility could be the use of linear logic, in which individual propositions can only be used once in any given proof, as a means of giving a semantics to consuming conditions, and various temporal logics (such as those surveyed in [Vila, 1995]) as a means of giving a semantics to time-outs. As Gabbay [Gabbay, 1996] discusses, resource logics like linear logic are captured naturally in systems of argumentation[1], and it is also natural to consider extending the predicates we use to have explicit temporal arguments.

[1] To be more precise Gabbay discusses how labelled deductive systems can be used to capture linear logic, but the necessary features of labelled deductive systems are shared with systems of argumentation.

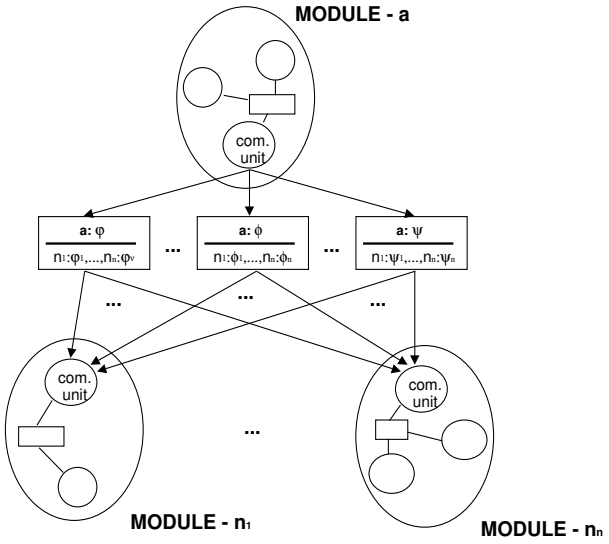

Figure 5.1: Module inter-connection (from a's perspective only)

5.3 Modular agents

Using units and bridge rules as the only structural elements is cumbersome when building complex agents (as can be seen from the model we developed in [Parsons et al., 1998] or the complexity that can achieve a single module like the one described in chapter 6). As the complexity of the agent increases, it rapidly becomes very difficult to deal with the necessary number of units and their interconnections using bridge rules alone. Adding new capabilities to the agent becomes a complex task in itself. To solve this problem we suggest adding another level of abstraction to the model—the *module*. Essentially we group related units into modules and separate interconnections into those inside modules and those between modules. This abstraction is, of course, one of the main conceptual advantages of object orientation [Booch, 1994].

5.3.1 Introducing modules

A module is a set of units and bridge rules that together model a particular capability or facet of an agent. For example, planning agents must be capable of managing resources, and such an agent might have a module modelling this ability. Similarly, such an agent might have a module for generating plans, a module for handling communication, and so on. Note that currently we do not allow modules to be nested inside one another, largely because we have not yet found it necessary to do so. However, it seems likely that we will need to develop a means of handling nested hierachies of modules in order

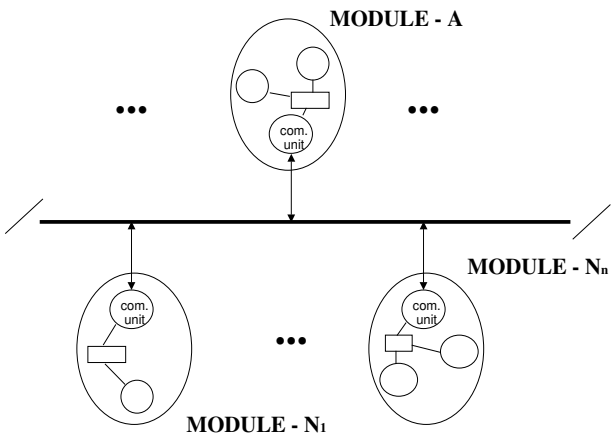

Figure 5.2: A pictorial explanation of the bus metaphor

to build more complex agents than we are currently constructing.

Each module must have a communication unit. This unit is the module's unique point of contact with the other modules and it knows what kind of messages its module can deal with. All of an agent's communication units are inter-connected with the others using *multicast bridge rules* (*MBR*s) as in Figure 5.1. This figure shows three MBRs (the rectangles in the middle of the diagram) each of which has a single premise in module a and a single conclusion in each of the modules n_i. The use of broadcast communication within the agent was chosen for convenience and simplicity—it is clearly not an essential part of the approach. It does, however, enhance the plug and play approach we are aiming for since when broadcast is used it is not necessary to alter the message handling within an agent when modules are added or removed.

Note that under this scheme *all* modules receive *all* messages, even those messages that are not specially for them. This obviates the need for a central control mechanism which routes messages or chooses which modules should respond to requests from other modules. With this type of connection, adding or removing a module doesn't affect the others (in a structural sense). We can see this communication net as a bus connecting all modules and firing a MBR is the same as putting a message onto this bus. There are as many kinds of messages running along this bus as there are MBRs (see Figure 5.2).

Since the MBRs send messages to more than one module, a single message can provoke more than one answer and, hence, contradictory information may appear. There are many possible ways of dealing with this problem, and here we consider one of them which we have found useful as an example. We associate a weight with each message. This value is assigned to the message by the communication unit of the module that sends it out. Weights are drawn from the interval $[0, 1]$ (maximum importance is 1 and minimum is 0), and their meaning is the strength of the opinion given in the message.

For example, in the multi-context version of the ReGreT model described in chapter 6, the weight of a message indicates the degree of reliability that has the trust value from the point of view of the module. These weights can also be used to resolve contradictory messages. For instance, the message with highest weight might be preferred, or the different weights of incoming messages could be combined by the communication unit receiving them to take a final decision (for instance using the belief revision mechanism described in [Parsons and Giorgini, 1999] where weights are taken to be degrees of belief in the truth of the message). Note that we only use weights in *inter module* messages, though it is quite possible to imagine using weights, especially those representing degress of belief, in *inter agent* messages as well.

Obviously, the use of modules does not solve every problem associated with altering the structure of an agent. For instance, if the only module which can perform a given task is removed, the agent will no longer be able to perform this task. Similarly, if one module depends on another module to do something and the second is removed, the first module becomes useless. However, the use of modules does simplify dealing with these kinds of interdependencies. Working at the unit level these problems are very difficult to deal with, but they became easier to handle when working at the module level because the number of modules that constitute an agent is usually small (7 or 8 in a relatively complex agent) and their area of influence is clearly circumscribed.

5.3.2 Messages between modules

We start with a set AN of agent names and a set MN of module names. Our convention is that agent names are upper case letters, and module names are lower case letters. An inter module message has the form:

$$I(S, R, \varphi, G, \psi)$$

where

- I is an illocutionary particle that specifies the kind of message. Examples of illocutions are ask, $query$, $inform$ or $answer$.
- S and R both have the form $A[/m]^*$[2] where $A \in AN$ or $A = Self$ (*Self* refers to the agent that owns the module) and $m \in MN$ or $m = all$ (*all* denotes all the modules within that agent). S reflects who is sending the message and R indicates to whom it is directed.
- φ is the content of the message.
- G is a record of the derivation of φ. It has the form: $\{\{\Gamma_1 \vdash \varphi_1\} \ldots \{\Gamma_n \vdash \varphi_n\}\}$ where Γ is a set of formulae and φ_i is a formula with $\varphi_n = \varphi$.
- $\psi \in [0, 1]$ is the weight associated with the message.

[2] As elsewhere we use BNF syntax, so that $A[/m]^*$ means A followed by one or more occurrences of $/m$.

Note that G is exactly the set of grounds of the argument for φ [Parsons et al., 1998]. Where the agent does not need to be able to justify its statements, this component of the message can be discarded. Note that, as argued by Gabbay [Gabbay, 1996] this approach is a generalisation of classical logic—there is nothing to stop the same approach being used when messages are just formulae in classical logic.

To see how this works in practice, consider the following. Suppose that an agent (named B) has four modules (a, b, c, d). Module a sends the message:

$$ask(Self/a, Self/all, Give(B, A, Nail), \psi_1, 0.5)$$

This means that module a of agent B is asking all the other modules in B whether B should give an agent called "A" a nail. The reason for doing this is ψ_1 and the weight a puts on this request is 0.5. Assume modules c and d send the answer

$$answer(Self/c, Self/a, not(Give(B, A, Nail)), \psi_2, 0.6)$$

and

$$answer(Self/d, Self/a, not(Give(B, A, Nail)), \psi_3, 0.7)$$

while module b sends

$$answer(Self/b, Self/a, Give(B, A, Nail), \psi_4, 0.3)$$

Currently we treat the weights of the messages as possibility measures [Dubois and Prade, 1988], and so combine the disjunctive support for $not(Give(B, A, Nail))$ using $\max$. As this combined weight is higher than the weight of the positive literal, the communication unit of Module a will accept the opinion *not (Give (B, A, Nail))*.

These degrees can change over time as the agent in question obtains new information which modifies its beliefs and intentions. In the last example, suppose that after some period of time, Module b receives new information which enables it to deduce that giving something to agent A is very desirable. It can increase the degree of importance of its answer:

$$answer(Self/b, Self/a, Give(b, a, Nail), \psi_4, 0.9)$$

Later, if Module a tries to give a Nail to agent A, it will succeed.

The messages we have discussed so far are those which are passed around the agent itself in order to exchange information between the modules which compose it. Our approach also admits the more common idea of messages *between* agents. Such inter agent messages have the same basic form, but they have two minor differences:

- S and R are agent names (i.e. $S, R \in AN$), no modules are specified.
- there is no degree of importance (because it is internal to a particular agent). However inter agent messages could be augmented with a degree of belief [Parsons and Giorgini, 1999] which could be based upon the weight of the relevant intra-agent messages. For instance, the witness information that receives an agent with a trust and reputation model like ReGreT should have associated a degree of reliability (assigned by the witness itself) that indicates how confident the witness about that information is.

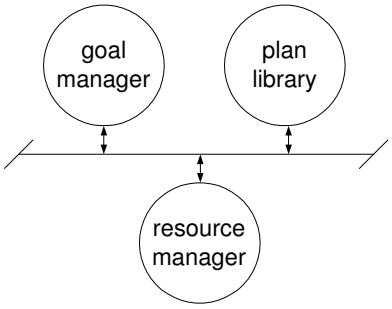

Figure 5.3: The modules in the agent

Thus, a message from B to A offering the Nail mentioned above would have the form:

$$inform(A, B, Give(B, A, Nail), \psi_5)$$

With this machinery in place, we are ready to specify realistic agent architectures.

5.4 Specifying a simple agent

This section gives a specification of a simple agent using the approach outlined above. The agent in question is a simple version of the home improvement agents first discussed in [Parsons and Jennings, 1996], which is supposed to roam the authors' homes making small changes to their environment. In particular the agent we discuss here attempts to hang pictures. As mentioned, the agent is rather simpler than those originally introduced, the simplification being intended to filter out unnecessary details that might confuse the reader. As a result, compared with the more complex versions of the home improvement agents described in [Parsons et al., 1998], the agent is not quite solipsistic (since it has some awareness of its environment) but it is certainly autistic (since it has no mechanisms for interacting with other agents). Subsequent sections build upon this basic definition to produce more sophisticated agents.

5.4.1 A high-level description

The basic structure of the agent is that of Figure 5.3. There are three modules connected by multicast bridge rules. These are the plan library (PL), the resource manager (RM), and the goal manager (GM). Broadly speaking, the plan library stores plans for the tasks that the agent knows how to complete, the resource manager keeps track of the resources available to the agent, and the goal manager relates the goals of the agent to the selection of appropriate plans.

There are two types of illocution which get passed along the multicast bridge rules. These are the following:

- *ask*: a request to another module.

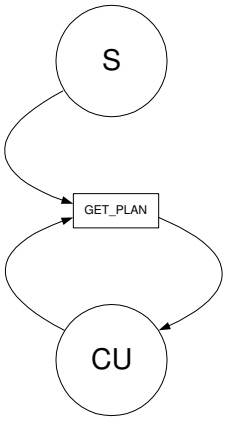

Figure 5.4: The plan library module (PL)

- $answer$: an answer to an inter module request.

Thus , what all the modules can do is to make requests to one another and answer those requests. We also need to define the predicates which form the content of such messages. Given a set of agent names AN, and with $AN' = AN \cup \{Self\}$.

- $goal(X)$: X is a string describing an action. This denotes the fact that the agent has the goal X.
- $have(X, Z)$: $X \in AN'$ is the name of an agent (here always instantiated to $Self$, the agent's name for itself, but a variable since the agent is aware that other agents may own things), and Z is the name of an object. This denotes Agent X has possession of Z.

Note that, from now on, we adopt a Prolog-like notation in which the upper case letters X, Y, Z, P are taken to be variables.

As can be seen from the above, the content of the messages is relatively simple, referring to goals that the agent has, and resources it possesses. Thus a typical message would be a request from the goal manager as to whether the agent possesses a hammer:

$$ask(Self/GM, Self/all, goal(have(Self, hammer)), \{\})$$

Note that in this message, as in all messages in the remainder of this chapter, we ignore the weight in the interests of clarity. Such a request might be generated when the goal manager is trying to ascertain if the agent can fulfill a possible plan which involves using a hammer.

5.4.2 Specifications of the modules

Having identified the structure of the agent in terms of modules, the next stage in the specification is to detail the internal structure of the modules in terms of the units they

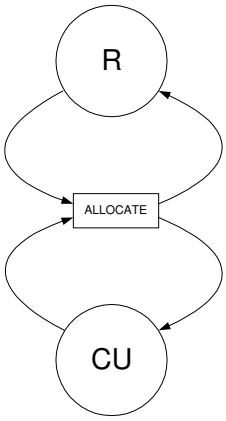

Figure 5.5: The resource manager module

contain, and the bridge rules connecting those units. The structure of the plan library module is given in Figure 5.4. In this diagram, units are represented as circles, and bridge rules as rectangles. Arrows into bridge rules indicate units which hold the antecedents of the bridge rules, and arrows out indicate the units which hold the consequents. The two units in the plan library module are:

- The communication unit (CU): the unit which handles communication with other units.
- The plan repository (S): a unit which holds a set of plans.

The bridge rule connecting these units is:

$$\mathsf{GET\_PLAN} \quad = \quad \frac{\begin{array}{c} CU > ask(Self/Sender, Self/all, goal(Z), \{\}), \\ S : plan(Z, P) \end{array}}{CU : answer(Self/PL, Self/Sender, goal(Z), \{P\})}$$

where the predicate $plan(Z, P)$ denotes the fact that P, taken to be a conjunction of terms, is a plan to achieve the goal Z[3].

When the communication unit sees a message on the inter module bus asking about the feasibility of the agent achieving a goal, then, if there is a plan to achieve that goal in the plan repository, that plan is sent to the module which asked the original question. Note that the bridge rule has a consuming condition—this is to ensure that the question is only answered once.

The structure of the resource manager module is given in Figure 5.5. The two units in this module are:

- The communication unit (CU).

[3]Though here we take a rather relaxed view of what constitutes a plan—our "plans" are little more than a set of pre-conditions for achieving the goal.

- The resource repository (R): a unit which holds the set of resources available to the agent.

The bridge rule connecting the two units is the following:

$$\textsf{ALLOCATE} \quad = \quad \frac{\begin{array}{c} CU > ask(Self/Sender, Self/RM, goal(have(Self, Z)), \{\}), \\ R > resource(Z, free) \end{array}}{\begin{array}{c} CU : answer(Self/RM, Self/Sender, have(Self, Z), \{\}), \\ R : resource(Z, allocated) \end{array}}$$

where the $resource(Z, allocated)$ denotes the fact that the resource Z is in use, and $resource(Z, free)$ denotes the fact that the resource Z is not in use.

In a similar way, it is clear that in a complete specification, a bridge rule to deallocate a resource would be necessary. We have obviated this bridge rule in the current specification because it is not relevant for our example. This help us to maintain things as simple as possible.

When the communication unit sees a message on the inter module bus asking if the agent has a resource, then, if that resource is in the resource repository and is currently free, the formula recording the free resource is deleted by the consuming condition, a new formula recording the fact that the resource is allocated is written to the repository, and a response is posted on the inter module bus. Note that designating a resource as "allocated" is not the same as consuming a resource (which would be denoted by the deletion of the resource), and that once again the bridge rule deletes the original message from the communication unit.

The goal manager is rather more complex than either of the previous modules we have discussed, as is immediately clear from Figure 5.6 which shows the units it contains, and the bridge rules which connect them. These units are:

- The communication unit (CU).
- The plan list unit (P): this contains a list of plans the execution of which is currently being monitored.
- The goal manager unit (G): this is the heart of the module, and ensures that the necessary sub-goaling is carried out.
- The resource list module (R): this contains a list of the resources being used as part of plans which are currently being executed.

The bridge rules relating these units are as follows. The first two bridge rules handle incoming information from the communication unit:

$$\textsf{RESOURCE} \quad = \quad \frac{CU > answer(Self/RM, Self/GM, have(Self, Z), \{\})}{R : Z}$$

$$\textsf{PLAN} \quad = \quad \frac{CU > answer(Self/PL, Self/GM, goal(Z), \{P\})}{P : plan(Z, P)}$$

The first of these, RESOURCE, looks for messages from the resource manager reporting that the agent has possession of some resource. When such a message arrives,

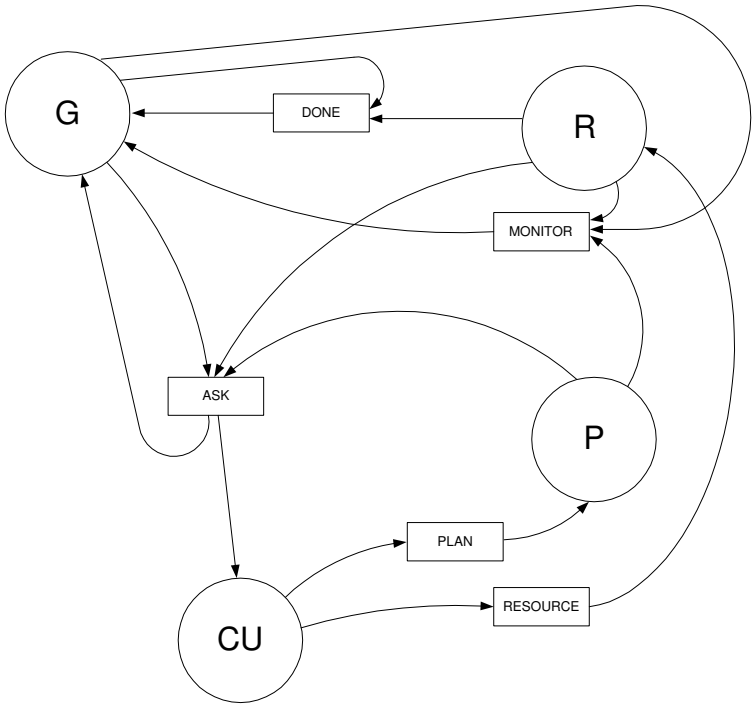

Figure 5.6: The goal manager module (GM)

the goal manager adds a formula representing the resource to its resource list module. The second bridge rule PLAN does much the same for messages from the plan library reporting the existence of a plan—such plans are written to the plan library. There is also a bridge rule ASK which generates messages for other modules:

$$\mathsf{ASK} \quad = \quad \frac{\begin{array}{c} G : goal(X), \\ G : not(done(X)), \\ R : not(X), \\ P : not(plan(X, Z)) \\ G : not(done(ask(X))), \end{array}}{\begin{array}{c} CU : ask(Self/G, Self/all, goal(X), \{\}), \\ G : done(ask(X)) \end{array}}$$

If the agent has the goal to achieve X, and X has not been achieved, nor is X an available resource (and therefore in the R unit), nor is there a plan to achieve X, and X has not already been requested from other modules, then X is requested from other modules and this request is recorded. Note that the "not" used in this bridge rule is not a logical negation but a negation by failure (like in PROLOG).

The remaining bridge rules are:

$$\textsf{MONITOR} \quad = \quad \frac{\begin{array}{c} G : goal(X), \\ R : not(X), \\ P : plan(X, P) \end{array}}{G : monitor(X, P)}$$

$$\textsf{DONE} \quad = \quad \frac{\begin{array}{c} G : goal(X), \\ R : X \end{array}}{G : done(X)}$$

The MONITOR bridge rule takes a goal X and, if there is no resource to achieve X but there is a plan to obtain the resource, adds the formula $monitor(X, P)$ to the G unit, which has the effect of beginnning the search for the resources to carry out the plan. The DONE bridge rule identifies that a goal X has been achieved when a suitable resource has been allocated.

5.4.3 Specifications of the units

Having identified the individual units within each module, and the bridge rules which connect the units, the next stage of the specification is to identify the logics present within the various units, and the theories which are written in those logics. For this agent most of the units are simple containers for atomic formulae. In contrast, the G unit contains a theory which controls the execution of plans. The relevant formulae are:

$$\begin{array}{rcl} monitor(X, P) & \rightarrow & assert\_subgoals(P) \\ monitor(X, P) & \rightarrow & prove(P) \\ monitor(X, P) \wedge proved(P) & \rightarrow & done(X) \\ \\ assert\_subgoals(\bigwedge_i Y_i) & \rightarrow & \bigwedge_i goal(Y_i) \\ prove(X \wedge \bigwedge_i Y_i) \wedge done(X) & \rightarrow & prove(\bigwedge_i Y_i) \\ \bigwedge_i done(Y_i) & \rightarrow & proved(\bigwedge_i Y_i) \end{array}$$

The *monitor* predicate forces all the conjuncts which make up its first argument to be goals (which will be monitored in turn), and kicks off the "proof" of the plan which is its second argument[4]. This plan will be a conjunction of actions, and as each is "done" (a state of affairs achieved through the allocation of resources by other bridge rules), the proof of the next conjunct is sought. When all have been "proved", the relevant goal is marked as completed.

[4]Given our relaxed view of planning, this "proof" consists of showing the pre-conditions of the plan can be met.

$ask(Self/GM, Self/all, goal(hangPicture(Self)), \{\})$	(GM1)
$answer(Self/PL, Self/GM, goal(hangPicture(Self)),$ $\{have(Self, picture) \wedge have(Self, nail) \wedge have(Self, hammer)\})$	(PL1)
$ask(Self/GM, Self/all, goal(have(Self, picture)), \{\})$	(GM2)
$ask(Self/GM, Self/all, goal(have(Self, nail)), \{\})$	(GM3)
$answer(Self/RM, Self/GM, have(Self, picture), \{\})$	(RM1)
$ask(Self/GM, Self/all, goal(have(Self, hammer)), \{\})$	(GM4)
$answer(Self/RM, Self/GM, have(Self, nail), \{\})$	(RM2)
$answer(Self/RM, Self/GM, have(Self, hammer), \{\})$	(RM3)

Table 5.1: The inter module messages

The specification as presented so far is generic—it is akin to a class description for a class of autistic home improvement agents. To get a specific agent we have to "program" it by giving it information about its initial state. For our particular example there is little such information, and we only need to add formulae to three units. The plan repository holds a plan for hanging pictures using hammers and nails:

$$\begin{aligned} S \quad : \quad & plan(hangPicture(X), \\ & have(X, picture) \wedge have(X, nail) \wedge have(X, hammer)) \end{aligned}$$

Of course, this is a very rudimentary plan, which only consists of the basic resources needed to achieve the goal of hanging a picture. The resource repository holds the information that the agent has a picture, nail and a hammer:

$$\begin{aligned} R \quad : \quad & resource(picture, free) \\ R \quad : \quad & resource(nail, free) \\ R \quad : \quad & resource(hammer, free) \end{aligned}$$

Finally, the goal manager contains the fact that the agent has the goal of hanging a picture:

$$G \quad : \quad goal(hangPicture(Self))$$

With this information, the specification is complete.

5.4.4 The agent in action

When the agent is instantiated with this information and executed, we get the following behaviour. The goal manager unit, which has the goal of hanging a picture, does not have the resources to hang the picture, and has no information on how to obtain them. It therefore fires the ASK bridge rule to ask other modules for input, sending message GM1 (see Table 5.1). When this message reaches the plan library, the bridge rule GET_PLAN is fired, returning a plan (PL1). This triggers the bridge rule PLAN in the goal manager, adding the plan to its P unit. This addition causes the MONITOR bridge

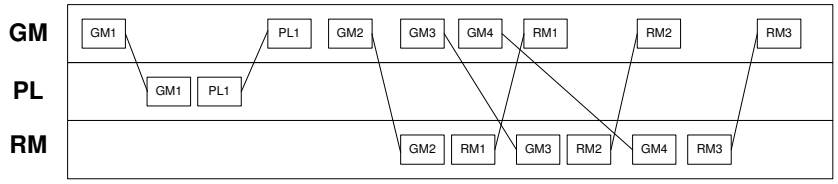

Figure 5.7: An execution trace for the agent

rule to fire. This, along with the theory in the G unit, causes the goal manager to realise that it needs a picture, hammer and nail, and to ask for these (GM2, GM3, GM4). As each of these messages reaches the resource manager, they cause the ALLOCATE rule to fire, identifying the resources as being allocated, and generating messages back to the goal manager (RM1, RM2, RM3). These resources cause the RESOURCE bridge rule in the goal manager to fire and the resources to be added to the resource list, R. The addition of the resouces is all that is required to complete the plan of hanging a picture, and the bridge rule DONE fires, adding the formulae $done(have(Self, picture))$, $done(have(Self, hammer))$ and $done(have(Self, nail))$ to the G unit. The theory in G then completes execution.

The messages passed between modules are represented in pictorial form in Figure 5.7—each row in the diagram identifies one module, time runs from left to right, and the diagonal lines represent the transfer of messages between modules.

5.5 Specifying more complex agents

This section gives a specification of a pair of agents which build upon those in the previous section. Indeed the agents introduced here are strict extensions of those in the previous section, containing all the components (down to the level of individual units) of the autistic agents and other components besides. The main extension is to reduce the autism of the model by giving each agent mechanisms for interacting with other agents.

5.5.1 A high-level description

The basic structure of the agent is that of Figure 5.8. There are four modules connected by multicast bridge rules. These are the plan library (PL), the resource manager (RM), the goal manager (GM) and the social manager (SM). The first three modules carry out the same basic functions as their namesakes in Section 5.4. The social manager handles interactions with other agents.

The intra-agent messages are exactly the same as for the autistic agent, but there are also two types of inter agent message, which broadly correspond to the ask and $answer$ messages. These are:

- $request$: a request to another agent.
- $reply$: an answer to an inter agent request.

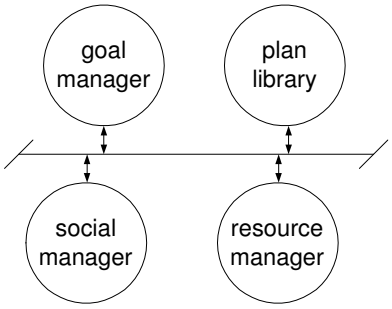

Figure 5.8: The modules in the agent

As in the previous section, uttering these illocutions are the only actions available to the agents. Thus the agents can talk about passing resources among them, but we provide no mechanisms for actually passing the resources.

The only new predicate which these agents employ is:

- $give(X, Y, Z)$: $X \in AN'$ and $Y \in AN'$ are agent names, and Z is a string describing a resource. This denotes X giving Z to Y.

and this is used in conjunction with the $request$ and $reply$ message types to build inter agent messages, which are of the form:

$$request(A, B, give(B, A, nail), \{\})$$

In this example, A requests that B gives a nail to A. In the following:

$$reply(B, A, give(B, A, nail), \{\})$$

B replies to A that B will give the nail to A.

5.5.2 Specifications of the modules

Once again, having decided on the overall structure of the agent, we have to specify the internal structure of the individual modules. The plan library module, pictured in Figure 5.9 has exactly the same structure as in the simple agent[5]. The two units in the plan library module are:

- The communication unit (CU): the unit which handles communication with other units.
- The plan repository (S): a unit which holds a set of plans.

[5]For ease of reference we repeat those parts of the specification which were also in the autistic agent model.

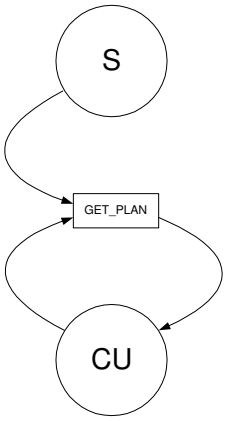

Figure 5.9: The plan library module (PL)

The bridge rule connecting these units is:

$$\mathsf{GET\_PLAN} \quad = \quad \frac{\begin{array}{c} CU > ask(Self/Sender, Self/all, goal(Z), \{\}), \\ S : plan(Z, P) \end{array}}{CU : answer(Self/PL, Self/Sender, goal(Z), \{P\})}$$

and the unit works in exactly the same way as in the simple agent.

The goal manager, pictured in Figure 5.10 is also the same as in the simple agent. The units are the following:

- The communication unit (CU).
- The plan list unit (P): this contains a list of plans the execution of which is currently being monitored.
- The goal manager unit (G): this is the heart of the module, and ensures that the necessary sub-goaling is carried out.
- The resource list module (R): this contains a list of the resources being used as part of plans which are currently being executed.

The bridge rules relating these units are:

$$\mathsf{RESOURCE} \quad = \quad \frac{CU > answer(Self/RM, Self/GM, have(Self, Z), \{\})}{R : Z}$$

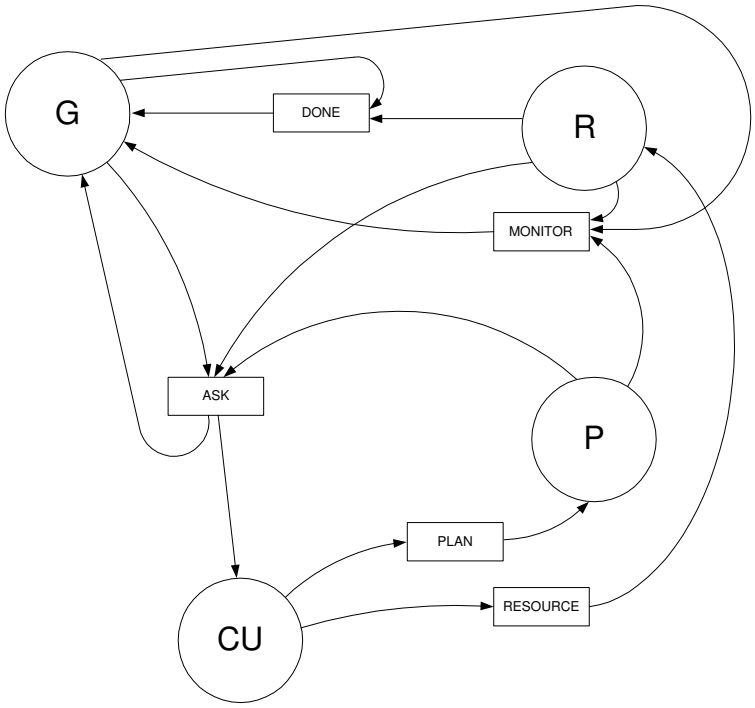

Figure 5.10: The goal manager module (GM)

$$\textsf{PLAN} \quad = \quad \frac{CU > answer(Self/PL, Self/GM, goal(Z), \{P\})}{P : plan(Z, P)}$$

$$\textsf{ASK} \quad = \quad \frac{\begin{array}{c} G : goal(X), \\ G : not(done(X)), \\ R : not(X), \\ P : not(plan(X, Z)), \\ G : not(done(ask(X))), \end{array}}{\begin{array}{l} CU : ask(Self/G, Self/all, goal(X), \{\}), \\ G : done(ask(X)) \end{array}}$$

$$\textsf{MONITOR} \quad = \quad \frac{\begin{array}{c} G : goal(X), \\ R : not(X), \\ P : plan(X, P) \end{array}}{G : monitor(X, P)}$$

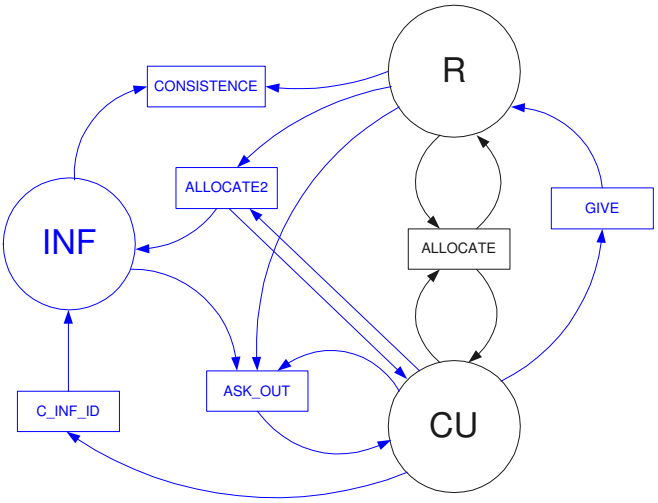

Figure 5.11: The resource manager module (RM)

$$\textsf{DONE} \quad = \quad \frac{\begin{array}{l} G : goal(X), \\ R : X \end{array}}{G : done(X)}$$

and these work in exactly the same way as in the simple agent. As can be seen by comparing Figure 5.5 with Figure 5.11, the resource manager of our new agents is considerably more complex than that in Section 5.4. This resource manager contains and extra unit INF which holds information about the resources possessed by agents in contrast with the R unit which simply records whether resources are free or allocated—this is a complication introduced by moving from one agent to several. Because the autistic agent does not deal with any external entities, any resources it considers belong to it, and any resources which do not belong to it do not exist as far as it is concerned. The social agents, in contrast, need to consider two aspects to every resource—whether or not it is free, and who has control over it. The R unit deals with the former, and the INF unit with the latter.

Clearly with more units we have more bridge rules. Of those in Figure 5.11, only the ALLOCATE rule is familiar from the autistic agent:

$$\textsf{ALLOCATE} \quad = \quad \frac{\begin{array}{l} CU > ask(Self/Sender, Self/Receiver, goal(have(X,Z)), \{P\}), \\ R > resource(Z, free) \end{array}}{\begin{array}{l} CU : answer(Self/RM, Self/Sender, have(X,Z), \{\}), \\ R : resource(Z, allocated) \end{array}}$$

where $resource(Z, allocated)$ denotes the fact that the resource Z is in use, and $resource(Z, free)$ denotes the fact that the resource Z is not in use. This rule will

be used if the agent is dealing with its own need for a resource that it owns, as in the case of the autistic agent.

Because we now have two agents, the resource that one agent requires may be owned by another agent, and this situation is where the INF unit comes into play. There are four bridge rules which relate this unit to R and CU. The first of these is C_INF_ID, which places knowledge about which agent has which resource into INF as a result of an $inform$ message:

$$\text{C\_INF\_ID} \quad = \quad \frac{CU > inform(U, V, have(X, Z), \{W\})}{INF : knowledge(have(X, Z))}$$

The name indicates that the rule is a kind of identity rule between the CU and INF units. Because in this model resources belong to just one agent, there is a contradiction if a resource is thought to belong two agents at once. The CONSISTENCE rule ensures that this situation does not occur by ensuring that the agent doesn't think another agent has a resource $knowledge(have(X, Z))$ when in fact the agent has the resource itself $resource(Z, free)$.

$$\text{CONSISTENCE} \quad = \quad \frac{\begin{array}{c} INF > knowledge(have(X, Z)), \\ R : resource(Z, free) \end{array}}{INF : knowledge(not(have(X, Z)))}$$

If an agent requires a resource it does not have, the ASK_OUT bridge rule allows it to request the resource from another agent, and the GIVE rule makes it possible to accept a resource if it is given:

$$\text{ASK\_OUT} \quad = \quad \frac{\begin{array}{c} CU : ask(Self/Sender, Self/Receiver, goal(have(X, Z)), \{P\}, \\ R : not(resource(Z, free)), \\ INF : knowledge(have(Y, Z)) \end{array}}{CU : ask(Self/RM, Self/SM, give(Y, X, Z, \{P\}}$$

$$\text{GIVE} \quad = \quad \frac{\begin{array}{c} CU > ask(Self/RM, Self/SM, give(X, Y, Z), \{P\}, \\ CU > answer(Self/SM, Self/RM, give(X, Y, Z), \{Q\} \end{array}}{R : resource(Z, free)}$$

The final resource-related situation an agent may be in is when it has a resource that another agent requires. This situation is handled by the ALLOCATE2 bridge rule, which hands over a resource if it is free and the social manager tells it to, updating the INF unit with information about where the resource is:

$$\text{ALLOCATE2} \quad = \quad \frac{\begin{array}{c} CU > ask(Self/SM, Self/Receiver, give(Self, Y, Z), \{P\}), \\ R > resource(Z, free) \end{array}}{\begin{array}{c} CU : answer(Self/RM, Self/SM, give(Self, Y, Z), \{P\}), \\ INF : knowledge(have(Y, Z)) \end{array}}$$

This completes the description of the resource manager.

The final module is the social manager. As can be seen from Figure 5.12, here the social manager consists of a single communication unit CU. This unit is responsible for receiving and sending *reply* and *request* messages from/to the social managers of the other agents. The generation of these messages is carried out by the theory in the communication unit.

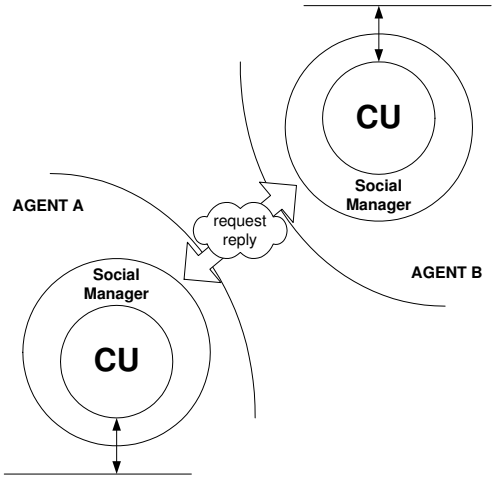

Figure 5.12: The social manager module (SM)

5.5.3 Specifications of the units

So far we have described the modules which make up the new agents, and for each module we have identified both the units which composed them and the connections necessary between the units. The next step is to decide what the internal structure of the units will be—which formulae and theories they will contain, and which logics those theories will be written in. Once again, there are not many units which include more than just a few atomic formulae. One of these is the unit G of the goal manager which contains the same theory as in the autistic agent:

$$\begin{aligned} monitor(X,P) &\rightarrow assert\_subgoals(P) \\ monitor(X,P) &\rightarrow prove(P) \\ monitor(X,P) \wedge proved(P) &\rightarrow done(X) \end{aligned}$$

$$assert\_subgoals(\bigwedge_i Y_i) \rightarrow \bigwedge_i goal(Y_i)$$

$$prove(X \wedge \bigwedge_i Y_i) \wedge done(X) \rightarrow prove(\bigwedge_i Y_i)$$

$$\bigwedge_i done(Y_i) \rightarrow proved(\bigwedge_i Y_i)$$

The other unit which contains more than just atomic formulae is the CU unit in the social manager, which contains:

$$ask(Self/Sender, Self/SM, give(X, Self, Z), \{\}) \rightarrow request(Self, X, give(X, Self, Z), \{\})$$

$$
\begin{aligned}
&reply(X, Self, give(X, self, Z), \{\}) \\
&\quad \wedge\, ask(Self/Sender, Self/SM, give(X, Self, Z), \{\}) \rightarrow \\
&\qquad\qquad answer(Self/SM, Self/Sender, give(X, Self, Z), \{\}) \\
&request(X, Self, give(Self, X, Z), \{\}) \rightarrow \\
&\qquad\qquad ask(Self/SM, Self/RM, give(Self, X, Z), \{\}) \\
&answer(Self/Sender, Self/SM, give(Self, X, Z), \{\}) \rightarrow \\
&\qquad\qquad reply(Self, X, give(Self, X, Z), \{\})
\end{aligned}
$$

This theory takes care of the translation from intra-agent messages to inter agent messages. The first formula takes an incoming ask message which contains a request for another agent X to give a resource, and converts it into a $request$ illocution. The second formula handles the reply to that request—if another agent responds positively to a request that the agent has previously made, then an $answer$ message is generated and sent to the originator of the request. The next two formulae handle responses to requests. The first of these takes a request for a resource from another agent and turns it into a message to the resource manager. The second takes a positive response, and converts that into a $reply$ message. The logic used in this unit, as in all the units in this agent specification, is classical first order logic.

As in Section 5.4, the specification up to this point is generic, defining something like a class description for simple non-autistic agents. For the particular scenario we have in mind, that of two agents which co-operate in hanging a picture, it is necessary to instantiate this generic description twice. The first instantiation creates an agent A which is virtually the same as the autistic agent of Section 5.4, the only difference being that A does not have the nail necessary to hang the picture, knowing instead that an agent B has the nail. A's plan repository holds the same plan as that of the autistic agent:

$$
\begin{aligned}
S \;:\; & plan(hangPicture(X), \\
& have(X, picture) \wedge have(X, nail) \wedge have(X, hammer))
\end{aligned}
$$

A's resource repository holds the information that the agent has a picture and a hammer:

$$
\begin{aligned}
R \;&:\; resource(picture, free) \\
R \;&:\; resource(hammer, free)
\end{aligned}
$$

while A's INF unit holds the information that B has a nail:

$$
\mathsf{INF} \;:\; knowledge(have(B, nail))
$$

Finally, A's goal manager contains the fact that the agent has the goal of hanging a picture:

$$
G \;:\; goal(hangPicture(A))
$$

This completes the specification of A. B is much simpler to instantiate, since it is only necessary to program it with the resource of a nail, by adding the following formula to

Agent A

$ask(Self/GM, Self/all, goal(hangPicture(A)), \{\})$	(GM1)
$answer(Self/PL, Self/GM, goal(hangPicture(A)),$ $\{have(A, picture) \wedge have(A, nail) \wedge have(A, hammer)\})$	(PL1)
$ask(Self/GM, Self/all, goal(have(A, picture)), \{\})$	(GM2)
$ask(Self/GM, Self/all, goal(have(A, nail)), \{\})$	(GM3)
$answer(Self/RM, Self/SM, give(B, A, nail), \{\})$	(RM1)
$ask(Self/GM, Self/all, goal(have(A, hammer)), \{\})$	(GM4)
$request(A, B, give(B, A, nail), \{\})$	(SM1)
$answer(Self/RM, Self/GM, have(A, picture), \{\})$	(RM2)
$answer(Self/RM, Self/GM, have(A, hammer), \{\})$	(RM3)
$answer(Self/SM, Self/RM, give(B, A, nail), \{\})$	(SM2)
$answer(Self/RM, Self/GM, have(A, nail), \{\})$	(RM4)

Agent B

$ask(Self/SM, Self/RM, give(B, A, nail), \{\})$	(SM1)
$answer(Self/RM, Self/SM, give(B, A, nail), \{\})$	(RM1)
$reply(B, A, give(B, A, nail), \{\})$	(SM2)

Table 5.2: The inter module messages

its resource repository:

$$R \quad : \quad resource(nail, free)$$

This completes the specification of the two agents.

5.5.4 The agents in action

If we execute these two agents, they generate and exchange the messages in Table 5.2 and Figure 5.13, which are very similar to those generated by the autistic agent. The main difference in this case concerns the provision of the nail required to hang the picture. In the case of the autistic agent, this nail was the property of the agent and so all the agent had to do to execute its "plan" of owning the nail was to allocate it. In this case when Agent A wants the nail it has to request it from B. Luckily for A, when B receives this request, it immediately agrees.

In more detail, the execution proceeds as follows. The goal manager unit of Agent A, has the goal of hanging a picture, does not have the resources to hang the picture, and has no information on how to obtain them. It therefore fires the ASK bridge rule to ask other modules for input, sending message GM1 (detailed in Table 5.1). When this message reaches A's plan library, the bridge rule GET_PLAN is fired, returning a plan (PL1). This triggers the bridge rule PLAN in the goal manager, adding the plan to its P unit. This addition causes the MONITOR bridge rule to fire. This, along with the

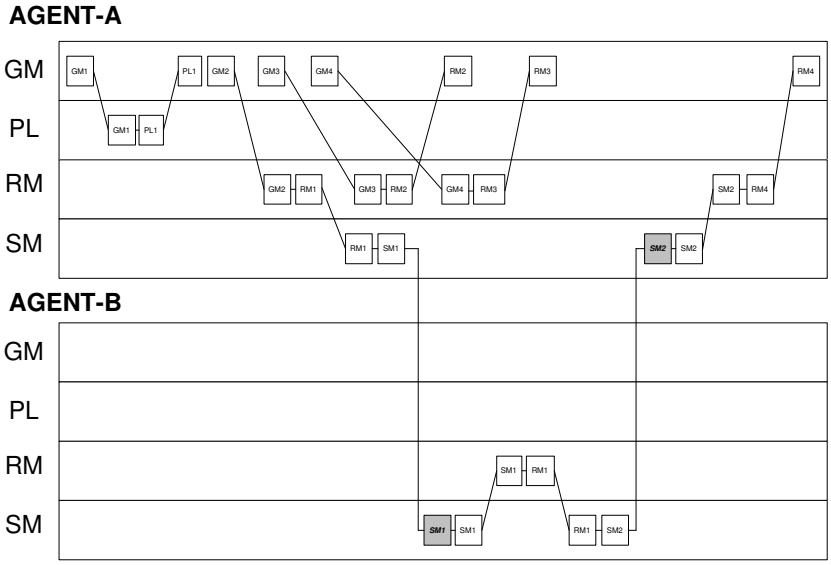

Figure 5.13: An execution trace for the agents

theory in the G unit, causes the goal manager to realise that it needs a picture, hammer and nail, and to ask for these (GM2, GM3, GM4). When GM2 and GM4 reach the resource manager, they cause the same sequence of events as in the autistic agent, firing the ALLOCATE rule, generating the messages RM2 and RM3 and allowing the goal manager to build part of its plan.

The problem, of course, is with GM3 which is requesting a nail. Since this is not a resource that A owns, the ASK_OUT rule is fired, generating RM1 which in turn sparks off activity in the social manager resulting in SM1. This $request$ is passed to B, where the social manager generates $SM1$. This goes to B's resource manager, triggering the ALLOCATE2 rule and then RM1 which confirms that B is happy to give a nail to A. The message passes back through B's social manager as SM2, is received by A social manager becoming A's SM2 message. This activates the GIVE rule in A's resource manager, which updates its resource list finally allowing it to alocate the nail. The message RM4 is sent to the goal manager which now has all the resources it requires. Consequently DONE fires, adding the formulae $done(have(Self, picture))$, $done(have(Self, hammer))$ and $done(have(Self, nail))$ to the G unit. The theory in G then completes execution.

5.6 Related Work

There are two main strands of work to which ours is related—work on executable agent architectures and work on multi-context systems. As mentioned above, most previous work which has produced formal models of agent architectures, for example dMARS [Ingrand et al., 1992], Agent0 [Shoham, 1993] and GRATE* [Jennings, 1995],

has failed to carry forward the clarity of the specification into the implementation—there is a leap of faith required between the two. Our work maintains a clear link between specification and implementation through the direct execution of the specification as exemplified in our running example. This relation to direct execution also distinguishes our work from that on modelling agents in Z [d'Inverno et al., 1998], since it is not yet possible to directly execute a Z specification[6].

More directly related to our work is that on DESIRE and Concurrent MetateM. DESIRE [Brazier et al., 1995, Treur, 1991] is a modelling framework originally conceived as a means of specifying complex knowledge-based systems. DESIRE views both the individual agents and the overall system as a compositional architecture. All functionality is designed as a series of interacting, task-based, hierarchically structured components. Though there are several differences, from the point of view of the proposal advocated in this paper we can see DESIRE's *tasks* as modules and *information links* as bridge rules. In our approach there is not an explicit task control knowledge of the kind found in DESIRE. There are no entities that control which units, bridge rules or modules are activated nor when and how they are activated. Also, in DESIRE the communication between tasks is carried out by the information links that are wired-in by the design engineer. Our inter module communication is organized as a bus and the independence between modules means new ones can be added without modifying the existing structures. Finally the communication model in DESIRE is based on a one-to-one connection between *tasks*, in a similar way to that in which we connect units inside a module. In contrast, our communication between modules is based on a multicast model. We could, of course, simulate the kind of control found in DESIRE by building a central controlling module, if this were required.

Concurrent MetateM defines concurrent semantics at the level of single rules [Fisher, 1998, Wooldridge, 1996]. An agent is basically a set of temporal rules which fire when their antecedents are satisfied. Our approach does not assume concurrency within the components of units, rather the units themselves are the concurrent components of our architectures. Our model has an inherent concurrent semantics at the level of the units and has no central control mechanism. Though our exemplar uses what is essentially first order logic (albeit a first order logic labelled with arguments), we could use any logic we choose—we are not restricted to a temporal logic as in MetateM.

There are also differences between our work and previous work on using multi-context systems to model agents' beliefs. In the latter [Giunchiglia, 1993], different units, all containing a belief predicate, are used to represent the beliefs of the agent and the beliefs of all the acquaintances of the agent. The nested beliefs of agents may lead to tree-like structures of such units (called *belief contexts*). Such structures have then been used to solve problems like the three wise men [Cimatti and Serafini, 1995]. In our case, however, any nested beliefs would typically be included in a single unit or module. Moreover we provide a more comprehensive formalisation of an autonomous agent in that we additionally show how capabilities other than that of reasoning about beliefs can be incorporated into the architecture.

[6]It is possible to animate specifications, which makes it possible to see what would happen if the specification were executed, but animating agent specifications is some way from providing operational agents of the kind possible using our approach.

Chapter 6

Engineering ReGreT using multi-context systems

6.1 Introduction

In the previous chapter we have shown how multi-context systems can be used to specify simple agents that can communicate to exchange information. In the two examples (see section 5.4 and section 5.5), for the sake of clarity, the complexity of the modules has been reduced to the minimum. In this section we want to show that multi-context systems are not restricted to simple, and somehow naive, specifications. We show how the multi-context approach can be used to specify ReGreT, the trust and reputation model presented in chapter 4. The result of this exercise is a multi-context module built using the constructs we have presented in the previous chapter and with the functionality of the trust and reputation model presented in chapter 4.

After an overview of the module in section 6.2 we present, grouped in several functional parts, the units and bridge rules that compound the multi-context version of the ReGreT system. In section 6.12 we propose operational semantics for the module. Finally, section 6.13 lists the set of bridge rules in the module and appendix A does the same for unit theories.

6.2 A high level description

The modular nature of the ReGreT system simplifies in great measure its specification using multi-context systems (at least at the module level). Figure 6.1 shows the structure of the ReGreT module.

The model has 16 units and 24 bridge rules. According to the complexity of the units we can establish the following classification:

- **Container units.** Units that are simple containers for facts. They use a first order logic and do not have an inference engine mechanism (*GR*, *IDB*, *SDB* and *CU*). This kind of units is not strictly necessary and could be removed without losing

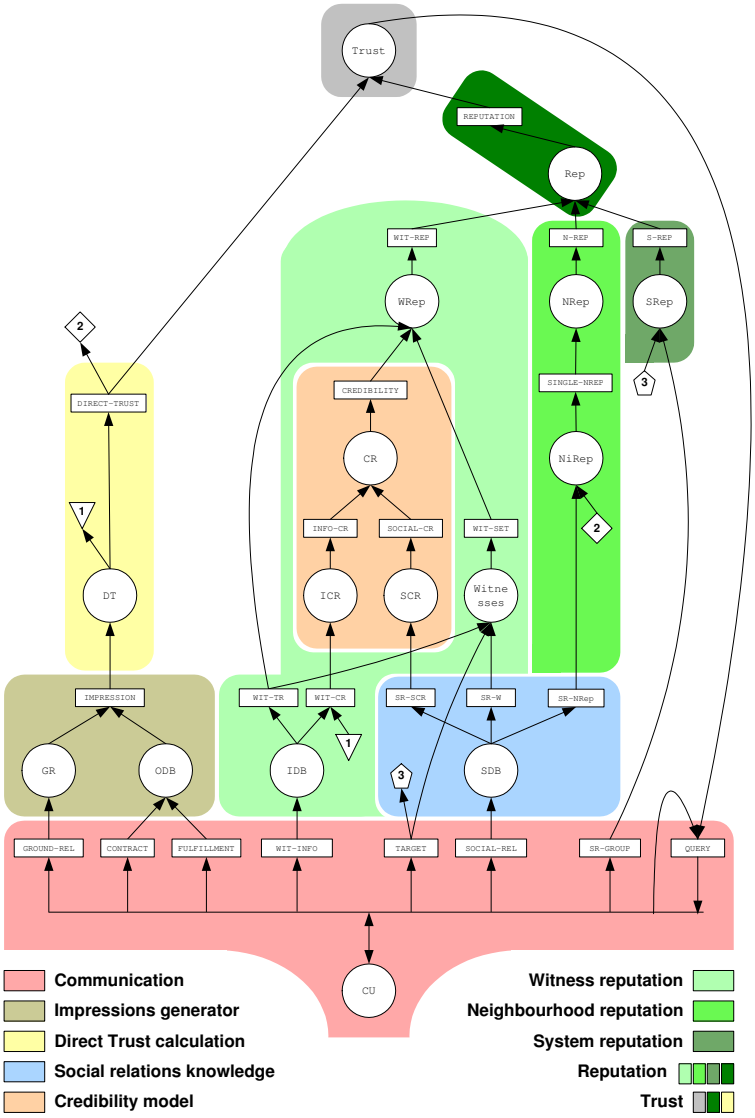

Figure 6.1: Multi-context specification of the ReGreT system

any functionality. However, they are essential to maintain a clear structure of the module and its different parts.

- **Aggregation units.** Units that aggregate values coming from other units and store the results as new facts (*DT*, *ICR*, *CR*, *WRep*, *NRep*, *Rep* and *Trust*).

- **Reasoning units.** Units with a reasoning mechanism. (*ODB*, *SCR*, *Witnesses*, *NiRep* and *SRep*).

On the other hand, bridge rules range from the simple "get here and put there" rule (that is, the rule gets something that has been deduced in one unit and introduces exactly the same into another) to the more complex bridge rules that have to make some calculation as part of the translation process (see for example the IMPRESSION bridge rule).

In order to facilitate the description of the module we will divide it in several functional blocks:

- **Communication**

- **Impressions generator**

- **Direct trust calculation**

- **Social relations knowledge**

- **Credibility model**

- **Witness reputation**

- **Neighbourhood reputation**

- **System reputation**

- **Reputation and trust**

With this division we also want to stress the intra-module organization of the units and bridge rules that, as we have said in the previous chapter, makes the multi-context approach a very good solution from a software engineering perspective. For example, if we want to change the way the agent generates impressions, it is easy to see that we have to change units *GR* and *ODB* and perhaps bridge rules IMPRESSION, GROUNDREL, CONTRACT and FULFILLMENT. What is important is that the links of this block with the rest of the elements that compound the module are clearly defined by the bridge rules and make the substitution a very simple and free of error process.

In the following sections we will go through each block analyzing the different units and bridge rules.

6.3 Communication

The *CU* is the link of the module with the rest of the elements that compound the agent. Together with the bridge rules attached to it, the communication unit feeds with new information several units and makes the work of the module available to other modules. The *CU* is quite simple. It uses a first order logic with no inference mechanism.

As we have said in chapter 5, an inter-module message has the form:

$$I(S, R, \varphi, G, \psi)$$

In the case of the ReGreT module, we have $I = \{ask, answer, inform\}$. At this moment, the module does not build (and cannot analyse) the derivation record of a trust value so the parameter G is not used neither for incoming nor outgoing messages.

The parameter ψ that weights the message is identified with the reliability of the content when the illocutionary particle is *Inform* or *Answer* and the urgency of the information when it is an *Ask*. For instance, in the message

$$answer(Self/ReGreT, Self/U, trust(Y, \phi, T), \_, 0.6)$$

the parameter ψ is indicating that the reliability of the trust value T associated to the behavioural aspect ϕ for agent Y is 0.6.

The set of bridge rules associated to incoming information (GROUND-REL, CONTRACT, FULFILLMENT, WIT-INFO, SOCIAL-REL and SR-GROUP) simply take the content of the inter-module message and add it into the corresponding unit theory (removing at the same time the message from the *CU*).

The QUERY bridge rule adds to the *CU* the answer to a request about the trust that a target agent deserves. It gets the information from the *Trust* unit. At the same time, it removes the query from the *CU* to be sure that the module answers the request only once. The *CU* "sends" the answer to the module that made the request. In the specification that we are presenting, the module is only prepared to respond to a request about the trust of a target agent. However it is easy, adding the corresponding bridge rules, to prepare the module to answer queries about credibility, reputation, etc.

Finally, the TARGET bridge rule is also fired when the module receives a request. This bridge rule adds a new target to the *Witnesses* and *SRep* unit theories.

6.4 Impressions generator

An *impression* in the ReGreT model can be defined as the impact (either positive or negative) that a contract and its fulfillment has in the agent's opinion about the partner. This part of the module generates these impressions.

Two units are used to model the impressions generator. A container unit (*GR*) that maintains the knowledge of the agent about the *grounding relations* and a unit that builds and stores the outcomes (*ODB*). Both units use first order logic.

Grounding relations are relations that link a behavioural aspect with a specific issue of an outcome and the function to evaluate that outcome. The predicate to represent a *grounding relation* is $gr(\phi, I_i, f)$ where ϕ is the behavioural aspect, I_i a specific issue

in the contract and f the function to evaluate an outcome (see chapter 4 for more details). For example, $gr(offers\_good\_quality, quality, f(X^c, X^s) = V(X^s) - V(X^c))$ informs that the issue in the contracts that has to be taken into account to measure if the partner offers good quality products is "quality" and the function to evaluate an outcome from that perspective is $f(X^c, X^s) = V(X^s) - V(X^c)$. The *GR* unit is a container for these predicates.

The *ODB* unit is responsible for generating outcomes from contracts and their fulfillments. The predicates for a contract and its fulfillment are the following:

$$contract(ID, Y, I, X^c, t) \quad fulfillment(ID, Y, I, X^f, t')$$

where *ID* is a unique identifier that links a contract with its fulfillment, Y is the partner for that contract, $I = \{I_1, \cdots, I_n\}$ an index of issues, X^c/X^f the value for the issues in the contract/fulfillment and t/t' a time stamp. The theory in the *ODB* unit is the single rule

$$contract(ID, Y, I, X^c, t) \wedge fulfillment(ID, Y, I, X^f, t') \rightarrow outcome(ID, Y, I, V^c, X^f, t')$$

and the inference mechanism Modus Ponens.

Besides the bridge rules that add new facts to the aforementioned units, there is only one more bridge rule related with the impressions generator part of the module. This bridge rule is what really makes the calculation of the impression.

The IMPRESSION bridge rule takes an outcome and a *grounding relation* (where the issue considered by the *grounding relation* is an issue that appears in the outcome), generates an impression and adds this impression to the *DT* unit's theory. The value of the impression is calculated by the bridge rule. Therefore, contrary to other bridge rules that only make a syntactic translation from one theory to another, in this bridge rule the translation implies the evaluation of a function.

6.5 Direct trust calculation

This part of the module is responsible for calculating the direct trust, that is, trust that is built only from direct interactions. The *DT* unit uses the impressions generated by the 'impressions generator block' we have explained in the previous section.

The method to calculate a direct trust and its reliability in the ReGreT model is explained in detail in chapter 4. Summarizing, the method consists on a weighted aggregation of impressions taking into account the recency of each impression. Given the mathematical nature of the method, we propose as the theory for this unit a logic program written in Prolog. Therefore, the inference mechanism will be resolution. The Prolog code is shown in A.2.

Note that the consequent of the IMPRESSION bridge rule is a Prolog *assert* that adds the impression directly to the *DT* unit's theory.

Finally, the DIRECT-TRUST bridge rule is a simple "get here and put there" rule.

6.6 Social relations knowledge

This block models the agent's knowledge about social relations in its environment. This knowledge is modeled using a set of *sociograms* (graph structures that show social relations), one for each relation type. It is composed by one unit (*SDB*) and three bridge rules (SR-SCR, SR-W and SR-NREP).

The *SDB* unit is a container unit with a first order logic and no inference mechanism. The predicates in this unit have the form $socialRel(RelType, X, Y, V)$, where $RelType$ is a social relation type between agents X and Y, and V is the intensity of that relation. Notice that these predicates are the edges of the *sociograms*. The knowledge stored in this unit is used in other three units. The task of bridge rules SR-SCR, SR-W and SR-NREP is to translate this knowledge into a format suitable for the logic in each one of these units.

Given that, the question is: Why do we need the *SDB* unit? There is no problem to connect bridge rules SR-SCR, SR-W and SR-NREP directly to the *CU*. However, we maintain the *SDB* unit to claim the importance of this block as a relevant and essential part of the module (a block that would disappear if we leave only the bridge rules).

It is true that, as it is, this block is only a container unit that stores social relations and a set of bridge rules that distribute this knowledge to other blocks. This is because we are supposing that the knowledge about social relations is available and ready to be used. However, in a real environment this will not be the case and this block should be replaced by another block that is able to build this knowledge from more basic information. So, although presently the *SDB* unit is just a container unit, it is likely to contain a more complex theory that would fully justify its presence.

It is important to remark again that thanks to the multi-context approach, this substitution (and similarly the substitution of the other blocks that compound the module) is a very clean and easy process.

6.7 Credibility model

The credibility model is composed of three units and the same number of bridge rules. This is the part of the module that evaluates the credibility of other agents when they are providing information (witnesses). Following the structure of the ReGreT system, two aspects are considered in order to determine the credibility of an agent. The analysis of previous information from that agent (modeled by unit *ICR* and bridge rule INFO-CR) and the social relations of that agent with the target and source agents (modeled by unit *SCR* and bridge rule SOCIAL-CR). Unit *CR*, appropriately aggregates both aspects in a single credibility value. Finally, bridge rule CREDIBILITY propagates this knowledge to other blocks.

The theory in the *ICR* unit is a logic program written in Prolog very similar to that used in the *DT* unit. You can find a detailed description of the method used to evaluate the credibility based on previous information in chapter 4. Basically, the agent compares previous pieces of information from that witness with its own perception about that information (based on direct experiences). Then, it performs a weighted aggregation of these accuracy values to obtain the credibility of the witness. Bridge

rule INFO-CR propagates these credibility values to the *CR* unit. The predicate added by the INFO-CR bridge rule to the *CR* unit's theory has the form $iCr(W, Cr, CrRL)$ where W is the witness ID, Cr the credibility value and $CrRL$ the reliability of that value.

As we have said, the second method that the ReGreT system uses to evaluate the credibility of a witness is the analysis of the social structure among that witness, the target agent and the source agent. The ReGreT system proposes the use of fuzzy rules to make this analysis. Given that, it is almost obvious the choice of a fuzzy logic with a forward chaining inference mechanism for unit *SCR* and a set of fuzzy rules as its theory. These fuzzy rules codify the necessary knowledge to analyse the aforementioned structure. The set of rules is the same we presented in chapter 4. To specify this unit we have used the *Milord II* language [Puyol-Gruart, 1996, Puyol-Gruart and Sierra, 1997, Puyol-Gruart et al., 1998, Godo et al., 2002]. *Milord II* is an architecture and a language for the development of knowledge-based systems. The language is based on modules as a method for programming in the large. Modules, generic modules and a set of operations among them are the basis of this language. A program in Milord II is then a hierarchical structure of modules. Modules are encapsulated components with a well defined interface. Each module is composed of deductive knowledge (facts and rules), local logic (an algebra declaration over uncertainty values) and a local control component (Horn-like meta-rules). The advantage of using this language to specify the unit is that we are approaching the specification to the implementation level.

As we will explain when we talk about the operational semantics in section 6.12, we want for our units an inference mechanism that works by saturation. In other words, we want an inference mechanism that once it is started tries to deduce as much as possible with the facts that are available in the unit's theory at that moment. Given the nature of the *Milord II* language we need an external inference control mechanism to obtain this behaviour.

We use Prolog to implement this mechanism. The Prolog part calls the *Milord II* engine with the different possibilities as parameters. The fuzzyfication and defuzzification of the values is performed by the *Milord II* engine. The Prolog code and the *Milord II* specification is shown in section A.4.

Similar to the INFO-CR bridge rule, the SOCIAL-CR bridge rule adds the credibility values calculated in the *SCR* unit to the *CR* unit. This time, the predicate added to the *CR* unit's theory has the form $sCr(W, T, Cr)$. This can be read as the credibility (Cr) that has witness W when is giving information about the target agent T.

Finally, *CR* is the unit responsible of aggregating both perspectives about the credibility. Like in the other aggregation units, we use a logic program written in Prolog. The code is shown in section A.5. The result of the inference process is a set of predicates with the form $wCr(W, T, Cr)$. The parameters have the same meaning as that for the social credibility but this time the credibility value Cr is a compendium of both, the social credibility and the credibility based on previous information. Again, you can find the details of this aggregation in chapter 4.

6.8 Witness reputation

The ReGreT system differentiates three types of reputation depending on the information source that is used to evaluate them. The witness reputation is the reputation that is calculated using the information gathered from other agents that belong to the community. The witness reputation block is composed by three units and four bridge rules.

The *IDB* unit is a container unit to store the opinions about trust received from witnesses. There are two bridge rules that propagate this knowledge.

The WIT-CR bridge rule adds new facts to theory in unit *ICR*. This bridge rule is fired only if (i) the agent has received information from a witness and (ii) there is a direct trust value that can be used to evaluate the accuracy of that information. Bridge rule WIT-TR, on the contrary, is fired each time a new piece of information appears in the *IDB* unit adding the relevant parts of it to the *WRep* and *Witnesses* unit theories.

The *Witnesses* unit is the unit that decides the witnesses that have to be taken into account to evaluate the reputation of a target agent. To build these lists the unit manages three information elements:

- The ID of the witnesses that have sent information and about whom is that information. This is received from the *IDB* unit through the WIT-TR bridge rule.
- The target agents. Received through the TARGET bridge rule.
- The social relations among agents. Received from the social relation knowledge block.

There are a lot of possibilities to make the selection of witnesses for a concrete target agent. One possibility, based on social network analysis, is presented as part of the ReGreT system in chapter 4, section 4.5.1. For this specification we will use a simpler approach. The set of witnesses that will be taken into account to calculate the witness reputation of a target agent Y on a concrete behavioural aspect ϕ will be simply the witnesses that have sent some information associated to that behavioural aspect for that specific agent and we know have had a trade relation with the target. This can be specified using first order logic and Modus Ponens as the inference mechanism. Then, a possible theory for unit *Witnesses* is:

$$\begin{aligned} target(X) \wedge witness(W, X, \phi) \wedge trade(W, X) &\rightarrow tmp(X, \phi, \emptyset) \\ witness(W, X, \phi) \wedge tmp(X, \phi, S) &\rightarrow tmp(X, \phi, S \cup W) \\ tmp(X, \phi, S) \wedge (\nexists witness(W, X, \phi) \wedge trade(W, X)) | W \in S) &\rightarrow wSet(X, \phi, S) \end{aligned}$$

The bridge rule WIT-SET is a "get here put there" bridge rule that takes the set of witnesses for each target agent calculated in unit *Witnesses* and adds them to theory of unit *WRep*.

Unit *WRep* calculates the witness reputation. The calculation is an aggregation of the opinions that a selected group of witnesses have provided. This selection is performed in unit *Witnesses*. The weight of each opinion is according to the credibility of each witness. The credibility model (see section 6.7) provides this knowledge. The logic program used to specify this unit is shown in section A.7.

Finally, bridge rule WIT-REP propagates witness reputation to the general reputation block.

6.9 Neighbourhood reputation

Neighbourhood reputation is the second reputation type that considers the ReGreT system. It takes into account only the social environment of the target agent and the kind of relations the target agent has established with that environment. The ReGreT system proposes the use of fuzzy rules to calculate this kind of reputation. This block is composed by two units and two bridge rules (see figure 6.1).

Unit *NiRep* is the kernel of this block and is very similar to unit *SCR*. It uses fuzzy logic and, as the *SCR* unit, is specified using the *Milord II* language and Prolog. This unit calculates a reputation value based on the social relations between each neighbour of the target agent and the target agent. It also calculates the reliability of that reputation value. The specification of unit *NiRep* is detailed in section A.8. Note that the Milord II part of the specification has two Milord modules, one for the reputation and the other for the reliability.

Bridge rule SINGLE-NREP adds the reputation due to each neighbour to the theory of unit *NRep*. Then, this unit aggregates these individual reputations into a single value. The Prolog code for this aggregation unit is shown in section A.9.

6.10 System reputation

In this section we will specify the last type considered in the ReGreT reputation model, the system reputation. This is the simplest and primary type of reputation an agent using the ReGreT system can calculate. It is based on the default reputation for an agent due to its ownership to a certain group. In the ReGreT system it is also taken into account the role the agent is playing. Here to simplify we only consider the group. Moreover we assume that an agent only belongs to a single group.

Bridge rule SR-GROUP adds to *SRep* unit's theory the information about which group each target agent belongs to. This information is acquired by the agent at run time. The predicate to represent this knowledge has the form $inGroup(Y, G)$.

The *SRep* unit uses a first order logic with Modus Ponens. Its initial theory is a set of predicates with the form

$$groupRep(G, \phi, Rep)$$

that codify the reputation (Rep) that has a group G (and therefore all its members) associated to behavioural aspect ϕ. As said in chapter 4, this information is usually inherited from relevant members of the group the agent belongs to. The theory of unit *SRep* is completed with the deduction rule:

$$target(Y) \wedge inGroup(Y, G) \wedge groupRep(G, \phi, Rep) \rightarrow sRep(Y, \phi, Rep)$$

Similar to bride rules WIT-REP and N-REP in the case of witness and neighbourhood reputation, bridge rule S-REP adds the system reputation values inferred in unit *SRep* to the *Rep* unit's theory.

6.11 Reputation and Trust

The reputation block is a composite block. It is formed by the witness, neighbourhood and system reputation blocks plus an aggregation unit (Rep) and a bridge rule (REPUTATION). The task of unit *Rep* is to aggregate the different reputation types into a single value. The aggregation is performed as described in chapter 4, section 4.5.4. Basically, the unit aggregates the witness, neighbourhood, system and default reputation types following this order of preference and considering the reliability associated to each reputation type value. The specification of this unit is detailed in section A.11. Note the *defaultRep(R,RL)* predicate denoting the default reputation of a target agent.

The REPUTATION bridge rule is a simple "get here put there" bridge rule that adds the reputation values inferred by unit *Rep* to the theory of unit *Trust*.

Finally, unit *Trust* is responsible of calculating the trust values that will be the outcome of the ReGreT module. This unit receives reputation values from the reputation block, and direct trust values from the direct trust calculation block. Then, according to the ReGreT system specification presented in chapter 4, it aggregates the values into a single trust value. The specification of this unit is detailed in section A.12.

6.12 Operational Semantic

In this section we propose an operational semantics for the ReGreT module. This operational semantics is also applicable to the examples in chapter 5.

As we have said, each unit has a different theory. The schema is always the same. Taking the unit as the center, we have a set of bridge rules that add new elements to this theory (what we call the input bridge rules) and a set of bridge rules that use the elements deduced by the theory (what we call the output bridge rules). The unique objective of the unit is to deduce things for the output bridge rules. Therefore, it seems coherent that the output bridge rules be the elements that lead the deduction process of the unit.

Each time a new fact is introduced in a unit theory by the input bridge rules, the antecedents of the output bridge rules are used as a guide to start the deduction process of the unit. The antecedents of the output bridge rules show what has to be deduced in the unit in order to apply the consequents. How the antecedents of the output bridge rules trigger the deduction process depends on the inference mechanism. Basically there are two possibilities: backward chaining and forward chaining inference mechanism.

If the unit has a backward chaining inference mechanism (like the one used by PROLOG), the antecedents of the output bridge rules are the query that starts the inference process. For instance, in those units that have a PROLOG program as theory, the antecedents of the output bridge rules specify the PROLOG predicate that is used to query the PROLOG engine. Take for example the unit *DT*. This unit has two output bridge rules: DIRECT-TRUST and WIT-CR. The antecedent in both bridge rules that refers to unit *DT* is $dt(Y, \phi, DT, DTRL)$ and this is exactly the main predicate of the PROLOG program that conforms the theory of unit *DT*. Each time a new predicate is added to unit *DT*, bridge rules DIRECT-TRUST and WIT-CR start the inference mechanism of unit *DT* by means of query '?- $dt(Y, \phi, DT, DTRL)$'. If the predicate can be proved

(and the other antecedents are satisfied), the bridge rule is fired.

If the unit has a forward chaining inference mechanism the only thing that the output bridge rules can do is to start the inference process of the unit and wait. One example of this type of unit is the Witnesses unit. Each time one of the (input) bridge rules WIT-TR, TARGET or SR-W add a new fact to the theory of unit Witnesses, (output) bridge rule WIT-SET starts the inference mechanism of the unit and waits until the inference process has finished. Then, it checks if there is any new fact that matches $wSet(X, \phi, \{s_1, \ldots, s_n\})$.

All new facts arrive through the communication unit (*CU*) and, like a wave, affect progressively different areas of the module. After some time, the units and bridge rules come back to a stable state where it is not possible to infer new things.

The module works as an any time algorithm. One can always query the module to obtain a result but this result is not necessarily the most updated one. For instance, suppose agent a receives a new fulfillment for a contract with agent b. This fulfillment is introduced into the ReGreT module and starts the process through the communication block, the impressions generator block and the direct trust calculation block to update the trust on agent b. However, at the same time, another module asks for the trust on agent b. Probably, the Trust unit has a trust value for agent b but it is not updated with the result of the last interaction. If the requirement is urgent, the module can send the current value in the Trust unit, otherwise it can wait until all the units and bridge rules come back to a stable state to be sure that the value is up-to-date.

This approach requires a control mechanism to monitor the state of the units and bridge rules.

Although in the ReGreT module the inference processes are very quick and therefore this control mechanism is not very relevant, you can have modules with units where the inference process is a matter of seconds, minutes or even hours. In these kind of modules is where this control mechanism becomes essential.

To finish this section we want to make a few comments about the connection between PROLOG and MILORD-II in units SCR and NiRep. Each MILORD-II module has a set of input variables (defined in the Import section of the module) and a set of output variables (defined in the Export section). The set of input variables are the variables that have to be instantiated before starting the inference process. Once the inference process comes to an end, the result is stored in the output variables.

The PROLOG part of the unit is responsible of instantiating the input variables of the MILORD-II module, start the inference mechanism and get the result from the output variables (in our case there is only a single output variable). The names of the PROLOG predicate that calls the MILORD-II module and the MILORD-II module name coincide as well as the name of the input and output variables.

6.13 Bridge rules

$$\text{GROUND} - \text{REL} = \frac{CU > Inform(Self/\_, Self/ReGreT, gr(\phi, I_i), \_, \_)}{GR : gr(\phi, I_i)}$$

$$\text{CONTRACT} = \frac{CU > Inform(Self/\_, Self/ReGreT, contract(ID, Y, I, X^c, t), \_, \_)}{ODB : contract(ID, Y, I, X^c, t)}$$

$$\text{FULFILLMENT} = \frac{CU > Inform(Self/\_, Self/ReGreT, fulfillment(ID, Y, I, X^f, t), \_, \_)}{ODB : fulfillment(ID, Y, I, X^f, t)}$$

$$\text{WIT} - \text{INFO} = \frac{CU > Inform(Self/\_, Self/ReGreT, wI(W, Y, \phi, T, TRL), \_, \_)}{IDB : wI(W, Y, \phi, T, TRL)}$$

$$\text{TARGET} = \frac{CU : Ask(Self/\_, Self/ReGreT, Trust(Y), \_, \_)}{Witnesses : target(Y), SRep : target(Y)}$$

$$\text{SOCIAL} - \text{REL} = \frac{CU > Inform(Self/\_, Self/ReGreT, socialRel(RelType, X, Y, V), \_, \_)}{SDB : socialRel(RelType, X, Y, V)}$$

$$\text{SR} - \text{GROUP} = \frac{CU > Inform(Self/U, Self/ReGreT, inGroup(X, G), \_, \_),}{SRep : inGroup(X, G)}$$

$$\text{QUERY} = \frac{\begin{array}{c} CU > Ask(Self/U, Self/ReGreT, Trust(Y), \_, \_), \\ Trust : trust(Y, \phi, T, TRL) \end{array}}{CU : Answer(Self/ReGreT, Self/U, trust(Y, \phi, T), \_, TRL)}$$

$$\text{IMPRESSION} = \frac{ODB : outcome(ID, Y, I, X^e, X^f, T)\ GR : gr(\phi, I_i, f), [I_i \in I]}{DT : assert(imp(ID, Y, \phi, v, T)) \quad \text{with } v = f(X^e, X^s)}$$

$$\text{DIRECT} - \text{TRUST} = \frac{DT : dt(Y, \phi, DT, DTRL)}{Trust : assert(dt(Y, \phi, DT, DTRL)), NiRep : assert(dt(Y, \phi, DT, DTRL))}$$

$$\text{SR} - \text{SCR} = \frac{SDB : socialRel(RelType, X, Y, V)}{SCR : assert(RelType(X, Y, V))}$$

$$\text{SR} - \text{W} = \frac{SDB : socialRel(RelType, X, Y, V)}{Witnesses : RelType(X, Y)}$$

$$\text{SR} - \text{NREP} = \frac{SDB : socialRel(RelType, X, Y, V)}{NiRep : assert(RelType(X, Y, V))}$$

$$\text{INFO} - \text{CR} = \frac{ICR : iCr(W, Cr, CrRL)}{CR : assert(iCr(W, Cr, CrRL))}$$

$$\text{SOCIAL} - \text{CR} = \frac{SCR : sCr(W, T, Cr)}{CR : assert(sCr(W, T, Cr))}$$

$$\text{CREDIBILITY} = \frac{CR : wCr(W, T, Cr)}{WRep : assert(cR(W, T, Cr))}$$

$$\text{WIT} - \text{CR} = \frac{IDB : wI(W, Y, \phi, T, TRL), DT : dt(Y, \phi, DT, DTRL)}{\begin{array}{c} ICR : assert(wI(W, Y, \phi, Comp)) \\ \text{with } Comp = (DTRL \cdot TRL \cdot Ap_0(T - DT)) \end{array}}$$

$$\text{WIT} - \text{TR} = \frac{\begin{array}{c} IDB : wI(W, Y, \phi, T, TRL), \\ WRep : assert(wI(W, Y, \phi, T, TRL)), Witnesses : witness(W, Y, \phi) \end{array}}{}$$

$$\text{WIT} - \text{SET} = \frac{Witnesses : wSet(X, \phi, \{s_1, \ldots, s_n\})}{WRep : assert(wSet(X, \phi, [s_1, \ldots, s_n]))}$$

$$\mathsf{WIT-REP} = \frac{WRep : wRep(Y, \phi, R, Rl)}{Rep : assert(wRep(Y, \phi, R, Rl))}$$

$$\mathsf{SINGLE-NREP} = \frac{NiRep : niRep(X, N, \phi, R, RL)}{NRep : assert(niRep(X, N, \phi, R, RL))}$$

$$\mathsf{N-REP} = \frac{NRep : nRep(Y, \phi, R, RL)}{Rep : assert(nRep(Y, \phi, R, RL))}$$

$$\mathsf{S-REP} = \frac{SRep : sRep(Y, \phi, R)}{Rep : assert(sRep(Y, \phi, R))}$$

$$\mathsf{REPUTATION} = \frac{Rep : reputation(X, \phi, R, Rl)}{Trust : assert(reputation(X, \phi, R, Rl))}$$

Chapter 7

Experiments

7.1 Introduction

In this chapter we will present a set of experiments to show how the ReGreT system behaves in different situations. We will present several scenarios to experiment with different parts of the system. The experiments are designed using the *SuppWorld* framework described in chapter 3 with some simplifications to make the results more comprehensible.

The set of experiments can be divided in two main blocks that correspond to two different scenarios. In the first scenario we consider a society with no other social relations among agents that trade relations. This scenario will allow us to test and show how the direct trust and the witness reputation work. In the second scenario we add cooperative and competitive relations among the members of the society and we analyze the ReGreT social credibility module.

7.2 The common framework

Although as we have said there are two different scenarios, both share a common structure. In this section we describe this structure.

The *Suppworld* base scenario used in all the experiments is composed by two grid markets. The first grid is the home of "producers" and the second grid the home of "manufacturers". The manufacturers buy products in the first market and then sell them in the second market. The buyers in the second market are simulated by a single entity that always buy all the product available from manufacturers. Figure 7.1 shows the sequence of events in the base scenario.

The first step is the entrance of rough material in the producers storage facilities. Then, during the production process scene the producers generate the product that will be sold to manufacturers in the market scene. Before starting the market scene, there is a convention of manufacturers that allow them to exchange information about producers. The market scene is the central event. Here the manufacturers negotiate with the producers to buy products. The round finishes with a production scene for manufacturers

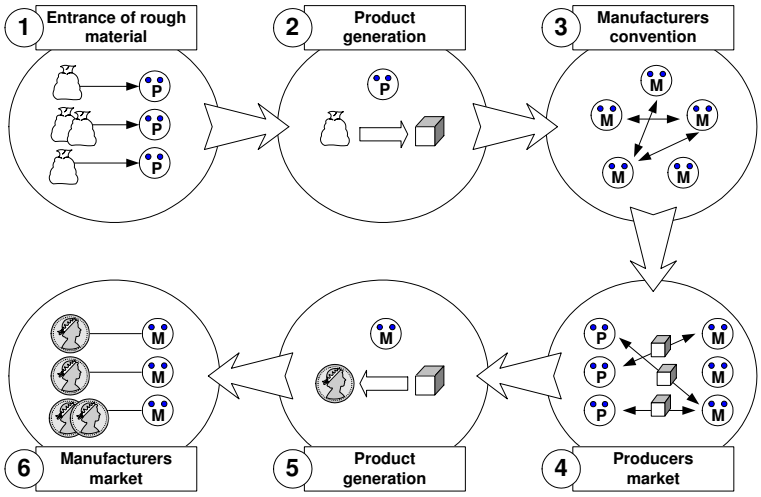

Figure 7.1: *SuppWorld* base scenario.

and the sale of the manufacturer product to the entity that simulates final consumers.

We have made some simplifications to the scenario in order to make the analysis of the results simpler and clearer.

Because we want to focus our analysis on the manufacturers, producers are designed to never lose money and to have always product in stock. Therefore, it never happens that a manufacturer cannot buy because the producer does not have products to sell. To ensure that, we give to the producers enough storage capacity to satisfy the demand, renewing this capacity each round without taking into account their cash.

Producers only sell one type of product. The presence of more than one product implies that manufacturers need some kind of strategy to adjust the amount of product of each type if they do not want to lose money (see section 3.6 in chapter 3). This strategy is very relevant for the final success of a buyer in a *Suppworld* scenario and, at this stage, complicates the analysis of trust and reputation mechanisms unnecessarily.

As we have said, the activity of producers is not taken into account in our experiments. Therefore, we have eliminated the possibility of performing coalitions and the capability of recommending other producers to manufacturers.

The last simplification is in the negotiation process. In a *SuppWorld* scenario, a contract has four issues: price, quantity, quality and transport type. Each issue has a minimum and maximum value that is fixed by the engineer for each experiment. This minimum and maximum values are used to ensure that agents are always negotiating within a controlled range for each issue. Since presently we do not want negotiation to interfere with the analysis of trust and reputation, all producers and manufacturers use the same negotiation engine with the same parameters. These parameters are adjusted to obtain the agreement exactly when the value for each issue is in the middle of its range. In our example this happens when *Price* = 30, *Quantity* = 10, *Quality* = 3 and *Transport_type* = 3. As we will see, this specific point in the negotiation has been chosen

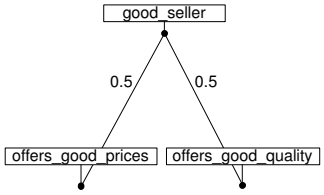

Figure 7.2: Ontological structure for a buyer.

to confer symmetry on the definition of good and bad behaviours when the agents fulfill a contract and it is not relevant in the analysis of trust and reputation.

Once the producer and the manufacturer arrive to an agreement, producer has to send the product to the manufacturer. At this point, producers can decide not to fulfill what was agreed. The quantity and the transport type are fixed. Therefore, the producer can only influence the price and the quality. This fact fixes another parameter that is common for all the experiments: the ontological structure of the buyers. This ontological structure is shown in figure 7.2 and defines a good seller as a seller that offers good prices and good quality, giving the same relevance to both aspects.

We differentiate five types of producers according to their behaviour in the fulfillment of contracts (what we have referred in chapter 3 as the "alignment" of the agent):

- SAINT: 30% of the times favours the partner by decreasing the price to the minimum and increasing the quality to the maximum.
- GOOD: 30% of the times favours the partner by decreasing the price and increasing the quality a quantity equal to $1/4$ of the issue's range.
- NEUTRAL: Always fulfill the contracts as agreed.
- BAD: 75% of the times cheats the partner by increasing the price and decreasing the quality a quantity equal to $1/4$ of the issue's range.
- EVIL: 75% of the times cheats the partner by increasing the price to the maximum and decreasing the quality to the minimum.

When they are not cheating/favouring the partner, all the producers fulfill the contracts as agreed.

Given that the agreement contract is fixed, it is possible to characterize the alignment of producers using the contract they send when cheat or favour the partner. Table 7.2 shows the relation between the alignment of a producer and the fulfillment of a contract (remember that we do not allow producers to modify the values for *Quantity* and *Transport_type*).

The position of manufacturers in the producers' grid at the beginning of each round is decided randomly. We have adopted this configuration, instead of considering their position in the home grid, to force the buyers to explore all the market and avoid the advantage of a good initial position. The movement of manufacturers is also random.

Alignment	Price	Quantity	Quality	Trans.Type
Agreement	30	10	3	3
Saint	10	10	5	3
Good	20	10	4	3
Neutral	30	10	3	3
Bad	40	10	2	3
Evil	50	10	1	3

Table 7.1: Relation between the alignment and the fulfillment of a contract.

To summarize, the base scenario for our experiments is a market grid populated by a fixed amount of sellers (producers) that offer all of them the same kind of product. At the beginning of each market session, a set of buyers (manufacturers) are distributed randomly over that grid. During the market session they move around randomly and buy things. However, sellers not always fulfill contracts as agreed. Manufacturers have only trust and reputation mechanisms to fight against cheaters. At the end of each round, manufacturers sell all the product to final consumers (simulated by a single entity). Depending on how each contract was fulfilled they will win or lose money in that transaction.

With the restrictions we have imposed, you can see the interaction among producers and manufacturers as a Prisoner's dilemma game where only producers decide if they want to cooperate (fulfill the contract as was agreed) or defect (cheat the partner by applying worse conditions in the fulfillment). Here, however, the producers can choose among different degrees of cooperation and defection.

7.3 Scenario I: Direct trust and witness reputation

The objective of these experiments is to see how an agent can improve its performance by using direct trust and witness reputation (without the credibility module). The performance of an agent is measured by the amount of cash that has won (or lost) after a fixed number of rounds.

We have compared four types of manufacturers:

- Manufacturers that always negotiate with producers. (Negotiate always)
- Manufacturers that use direct trust to decide if it is worth it or not to negotiate. (DT)
- Manufacturers that use direct trust and witness reputation. (DT + WR)
- Manufacturers that use direct trust and witness reputation but always provide wrong information to other manufacturers. (DT + WR (LIERS))

The first three types are tested in an environment where manufacturers always say the truth and do not try to lie the others. The fourth type, however, reproduces a situation where manufacturers always give wrong information.

We performed an initial set of experiments where the full population of manufacturers had, in turn, each one of the profiles commented before. We were expecting an increase of performance for the first three types of manufacturers but when we arrived at the manufacturers that were using witness reputation, we found that their performance was worse than the performance of manufacturers that were using only direct trust.

The reason for that is the nature of witness reputation. Suppose the situation where all the manufacturers start from scratch and witnesses only give information based on their own direct experiences. Even assuming the witnesses always tell the truth, a manufacturer only can provide reliable information after several rounds. For that time, however, all the other manufacturers have had a similar number of direct experiences and witness information is useless. In this scenario, witness reputation is redundant. Moreover, if producers do not have a fixed behaviour (that is, bad agents not always behave badly) you can have witnesses that are distributing wrong information. In that case, the use of witness information is self-defeating. This is what was happening in our experiments.

As you can imagine, the situation is even worse if witnesses also spread opinions based on reputation.

The conclusion is that witness reputation is only useful if you can guarantee that there is a pool of individuals that can provide well founded information. It is not worth it (and can be even self-defeating) the use of witness reputation in those situations where there is a general lack of knowledge.

Knowing that, we repeated the experiments but this time using the following procedure. All manufacturers use only direct trust for a number of rounds until we are sure that the general knowledge about the market among manufacturers is good enough. Then we selected a set of manufacturers and initialize their internal status (as if they were just arriving at the market), changing the parameters of the ReGreT system according to the characteristics we want to study. This is the moment when the experiment really starts. We monitor the performance of these selected set of manufacturers during a fixed number of rounds.

The main parameters of the experiments are the following:

- 16 producers (4x4 grid).
- 64 manufacturers (8x8 grid).
- 50 rounds. Each round has 30 ticks.
- From the 64 manufacturers, 10 are randomly selected to be monitored as explained before.
- 30 rounds preparing the market, 20 rounds monitoring the selected group of manufacturers.
- For each round there is a 1 tick convention of manufacturers.

Maintaining this configuration we made 5 sets of experiments, each set corresponding to a different alignment configuration for producers. Concretely we tried the following alignment configurations:

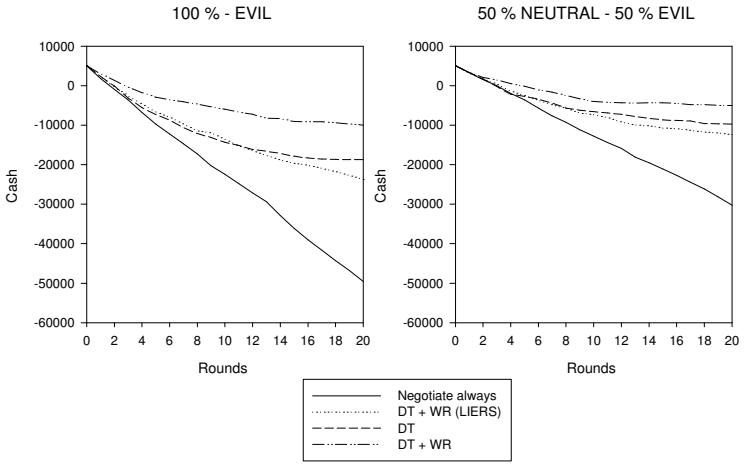

Figure 7.3: DT and WR experiments (I).

- 100% of producers are EVIL.
- 50% are BAD, 50% are EVIL.
- 50% are NEUTRAL, 50% are EVIL.
- 100% are BAD.
- 50% NEUTRAL, 30% BAD and 20% EVIL.

We tested each alignment configuration with the different type of manufacturers presented before, that is, manufacturers that always negotiate with producers, manufacturers that use direct trust to decide if it is worth it or not to negotiate, manufacturers that use direct trust and witness reputation and manufacturers that use direct trust and witness reputation but in an environment where witness information is always wrong.

The results of these experiments are shown in figure 7.3 and figure 7.4. These plots show, for each alignment configuration, the average performance of the selected manufacturers from the moment they are initialized to the end of the experiment.

This time, for all the alignment configurations, we get the expected result. There is a great improvement in the performance of manufacturers that use direct trust to decide if it is worth it or not to negotiate with a specific producer compared with the manufacturers that always negotiate. As it would be expected, this difference is bigger in hostile environments like the first three but also quite significant in more normal environments like the latest shown in figure 7.4. Also remarkable is the improvement of performance for those manufacturers that were using witness reputation in a collaborative environment compared with manufacturers that only were using direct trust.

Finally, you can observe that the performance of agents that use witness reputation in an environment where witness information is always wrong is always worse than the

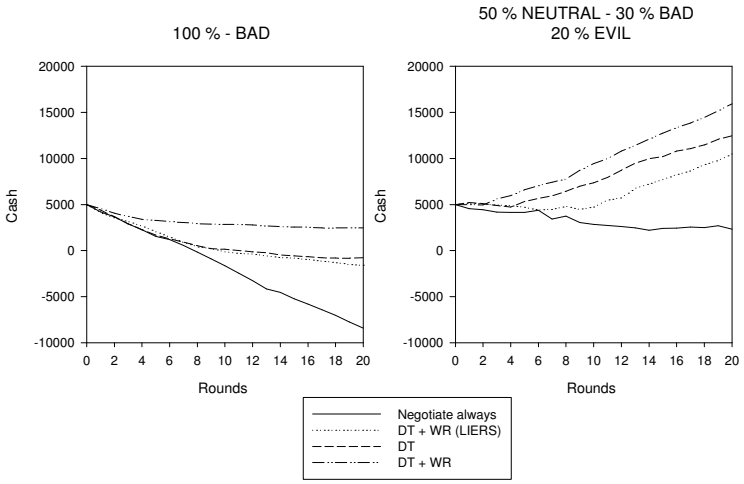

Figure 7.4: DT and WR experiments (II).

performance of manufacturers that use only direct trust. This supports our hypothesis that if there are no guarantees that witness information has a minimum quality it is better to ignore it. These results also validate the results obtained in the experiments with manufacturers that use witness reputation in a collaborative environment.

In order to check if these results can be extended to bigger societies we repeated the last experiment in a bigger scenario. The relevant parameters for this experiment are:

- 64 producers (8x8 grid).
- 256 manufacturers (16x16 grid).
- 90 rounds. Each round has 60 ticks.
- From the 256 manufacturers, 10 are randomly selected to be monitored as explained before.
- 50 rounds preparing the market, 40 rounds monitoring the selected group of manufacturers.
- For each round there is a 20 ticks convention of manufacturers.

The results are shown in figure 7.5. As you can observe, the results are very similar to those obtained for the medium size environment.

The complete set of parameters used in these experiments are detailed in appendix A.

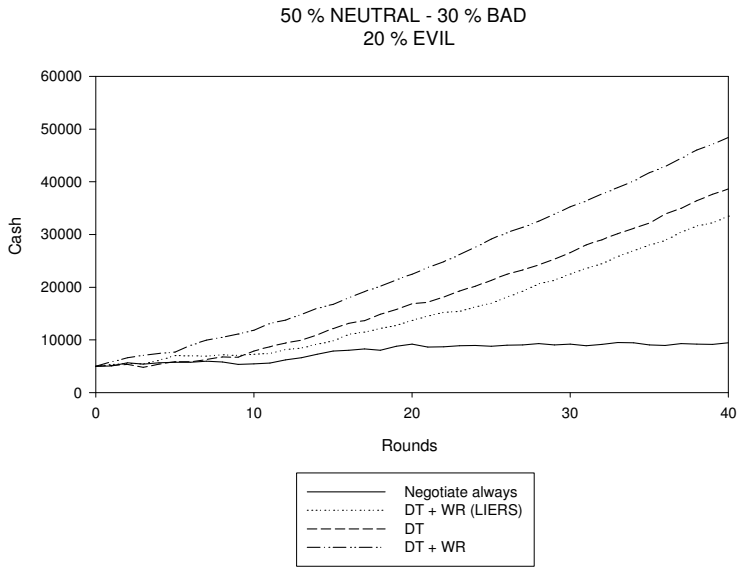

Figure 7.5: Large scenario.

7.4 Scenario II: Social credibility

In the scenario proposed in this section, we will add competitive and cooperative relations among the members of the society. Competitive and cooperative relations in a *SuppWorld* scenario (as in real life) have a direct impact on the behaviour of the agents. Because we want to analyse the social credibility module, we will focus on the impact that cooperative and competitive relations have in the truthfulness of the information given by witnesses.

We want to observe how the use of the social credibility module of the ReGreT system can improve the performance of the manufacturers in such kind of scenarios.

In a *SuppWorld* scenario, competitive and cooperative relations define the probability to deliberately provide wrong information. To compute this probability, an agent takes into account the competitive and cooperative relations among the agent that is making the query (the source), the subject of the query (the target) and the witness (the agent itself). To do that, it uses the same set of fuzzy rules used to calculate the social credibility (see table 4.1). In those cases where there is no cooperative and competitive relation among the source, the target and the witness, the witness computes the probability to cheat according to its alignment.

For this experiment we have compared three types of manufacturers:

- Manufacturers that use direct trust to decide if it is worth it or not to negotiate. (DT)
- Manufacturers that use direct trust and witness reputation. (DT + WR)

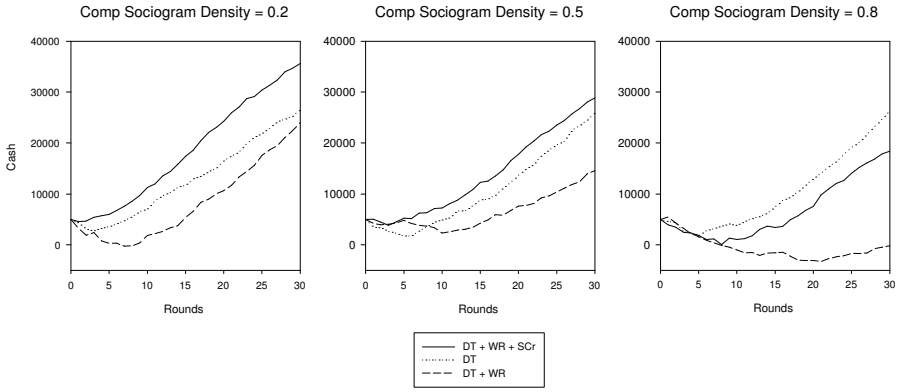

Figure 7.6: Social credibility and competitive relations.

- Manufacturers that use direct trust, witness reputation and the social credibility module. (DT + WR + SCr)

We have to say that the manufacturers that use the social credibility module have a perfect knowledge of the society. In a real world, the sociograms usually have some mistakes that would decrease the performance of the module.

The main parameters of the experiments are the following:

- 16 producers (4x4 grid).
- 64 manufacturers (8x8 grid).
- 50 rounds. Each round has 30 ticks.
- From the 64 manufacturers, 10 are randomly selected to be monitored.
- 20 rounds preparing the market, 30 rounds monitoring the selected group of manufacturers.
- For each round there is a 10 tick convention of manufacturers.

We maintain a fixed configuration of 25% SAINTs and 75% EVILs in the case of producers and a 100% of SAINTs in the case of manufacturers (thinking that the behaviour of the manufacturers is also conditioned by the social relations).

In the first set of experiments shown in figure 7.6 we can see the performance of the manufacturers given different environments, each one more competitive than the previous. The degree of competition in the environment is measured by the density of the competitive sociogram shared by all the manufacturers. For instance, a density of 0.5 means that a manufacturer has a competitive relation with 50% of the agents (manufacturers and producers) in that society.

It is interesting to see that even with a density of only 0.2, the performance of manufacturers that use witness reputation and direct trust is worse than the performance

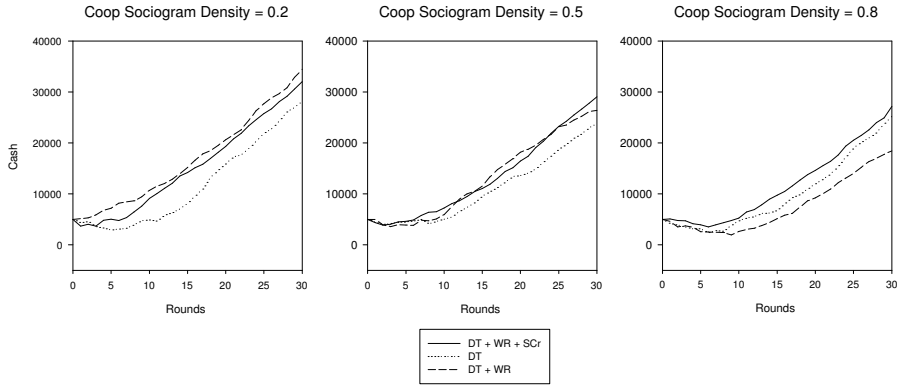

Figure 7.7: Social credibility and cooperative relations.

of those that use only direct trust. As expected, the social credibility module improves considerably the performance of the manufacturers that use witness reputation.

With a density of 0.5 the situation is similar. However the performance of the manufacturers using witness reputation (with or without the social credibility module) is worse than in the previous case (density = 0.2). This happens because by increasing the number of agents with a competitive relation we are decreasing the number of reliable witnesses and, as a consequence, we are decreasing the amount of useful information that can be used by the witness reputation module. In the last experiment we force this situation by fixing a density of 0.8. Here, even using the credibility module, the use of witness reputation is not worth it. Again, as we observed in the *Scenario I*, not always the use of witness reputation implies an improvement in the performance.

We repeated the same set of experiments with the same parameters but this time considering only cooperative relations. Figure 7.7 shows the obtained results.

It seems that cooperative relations (as they are defined in the *SuppWorld* framework) do not have the same relevance that competitive relations. The performance of the agents that use the social credibility module is very similar in the three experiments. The agents that use witness reputation without the social credibility module experiment a decrease in performance when the number of cooperative relations increases but it is not comparable with the decrease of performance due to competitive relations.

7.5 Conclusions

The aim of the experiments we have presented is to show how the *SuppWorld* framework can be used to test the ReGreT system and, in general, any complex trust and reputation model. These experiments are a small sample of the full set of experiments that are necessary to validate a system with the complexity of ReGreT. However, even knowing they are not complete, it is possible to extract some interesting conclusions.

As we have seen in the experiments of the *Scenario I* not always the use of witness

reputation contributes to improve the performance of the agent. There are situations where the use of witness reputation is self-defeating. Something similar happens with the use of social information.

One of the most important factors to be taken into account to decide if it is worth it or not the use of alternative sources of information is the cost to obtain direct experiences. In a scenario where it is easy and cheap to get direct experiences it is difficult that the use of other solutions to compute trust and reputation values compensates the problems associated to them. However, as long as the cost of direct experiences increases, the use of other sources of information like witness information or social knowledge become more and more important.

Given that, it seems clear that we cannot rely on a static mechanism to combine the different sources of information. In chapter 4 we proposed a method to combine the different sources of information of the ReGreT system. We are not saying that that method is worthless but that it only covers part of the possible scenarios. So if we really want a generic trust and reputation system we have to go for a dynamic method to combine the different sources of information that can adapt to the characteristics of the environment.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

8.1.1 Trust and reputation

In this thesis we have presented ReGreT. ReGreT is a trust and reputation system to be used in complex agent societies where the social relations among members play an important role.

The main characteristic of the ReGreT system is the use of several sources of information (direct experiences, information from third party agents and social structures) to calculate trust, reputation and credibility values instead of relying on a single source of information like other systems do. The combination of complementary methods that use different aspects of the interaction and social relations, allows the agent to calculate trust and reputation values at different stages of its knowledge of the society.

Associated to each trust or reputation value, the system calculates a measure of its reliability. This measure can be very useful in order to decide to which extend an agent can rely on the trust and reputation values to make a decision.

Contrary to other systems, ReGreT makes a clear distinction between trust and reputation. In our system, reputation is used to improve the calculation of trust values.

The ReGreT system also has a hierarchical ontology structure that allows us to consider at the same time several trust and reputation values associated to different behavioural aspects. Moreover, we consider that trust and reputation are not single and abstract concepts but rather multi-facet concepts. The *ontological structure* provides the necessary information to combine reputation and trust values linked to simple aspects in order to calculate values associated to more complex attributes.

Finally, we have shown how social network analysis can be used to improve the accuracy of the reputation values and, as a consequence, the trust values. Social network analysis is also used in the mechanism to select the best possible witnesses and in the credibility module.

We have also introduced the *SuppWorld* framework, an experimental framework designed to test trust and reputation models. This framework allows us to build simple scenarios oriented to test the basic aspects of trust and reputation models and also com-

plex scenarios where social relations acquire a great relevance. To show how it works we have designed a set of experiments to test several features of the ReGreT system.

The main conclusion extracted from these experiments is that although the use of witness reputation and social knowledge can improve spectacularly the performance of the agent, in some situations they are self-defeating. If we really want a generic trust and reputation system that can be used in a broad range of situations we have to go for a dynamic method to combine the different sources of information that can adapt to the characteristics of the environment.

8.1.2 Multi-context systems

We have proposed a general approach to defining agent architectures which extends the work of [Parsons et al., 1998, Sabater et al., 1999] with the idea of modules and, as a result, links the approach more strongly with the software engineering tradition. This approach provides a means of structuring logical specifications of agents in a way which makes them directly executable. The approach has a number of advantages over other work on defining agent architectures. Firstly it bridges the gap between the specification of agents and the programs which implement those specifications. Secondly, the modularity of the approach makes it easier to build agents which are capable of carrying out complex tasks such as distributed planning. From a software engineering point of view, the approach leads to architectures which are easily expandable, and have re-usable components.

From this latter point of view, our approach suggests a methodology for building agents which has similarities with object-oriented design [Booch, 1994]. The notion of inheritance can be applied to groups of units and bridge rules, modules and even complete agents. These elements could have a general design which is specialized to different and more concrete instances by adding units and modules, or by refining the theories inside the units of a generic agent template.

Then we have used the multi-context systems approach to specify ReGreT, the trust and reputation system presented in chapter 4. With this exercise we have demonstrated that, using multi-context systems with the extensions presented in chapter 5, it is possible to specify complex agents with advanced capabilities.

We have to recognize that the mathematical nature of the ReGreT system limits somehow the real potential of the multi-context systems approach. As we commented in chapter 5, multi-context systems are most appropriate when building agents which are logic-based and are therefore largely deliberative. Therefore, thinking in the area of trust and reputation models, we can adventure that the multi-context systems approach will be even more suitable for the specification of cognitive models.

8.2 Future work

We have divided the future work in two different parts. The future work related with trust and reputation models and the future work related with the design of autonomous agents using a multi-context approach.

8.2.1 Trust and reputation

The area of trust and reputation models for virtual societies is a very young, although very active, discipline and therefore there is still a lot of work to be done. Here we propose a short list of topics we think deserve a special attention:

- As stated in chapter 2, it is necessary to propose a common method to evaluate the different trust and reputation models. The number of models is increasing quickly and because different groups use different methods to evaluate their models, it is very difficult to establish a comparison. In chapter 3 we have presented the *SuppWorld*, a framework designed to evaluate trust, reputation, and negotiation models. This framework is a small step (we think in the right direction) toward the objective of having a common method to evaluate trust and reputation models. As a result of our work in the *SuppWorld* framework we have sketched the characteristics this method should have:

  - It has to be accepted by the community members. This is, without any doubt, the most important characteristic. You can have the most fantastic test-bed in the world but, if nobody is using it, it is useless.
  - You have to be able to evaluate from the most simple trust and reputation model to the most sophisticated. To achieve that, we propose an approach used in other areas of computer science. Instead of having a single test, we propose to have a set of them. Each individual test covers a different problem and scenario, associated to trust and reputation, with an increasing complexity. Usually, each model will be able to execute only a subset of these tests.
  - The time and effort necessary to adapt the models to run the test-bed has to be reduced to the minimum. Therefore, the link with the framework that supports the test-bed needs to be clean and, if it is possible, independent of the programming language that has been used to implement the models.
  - The test-bed has to be open. This means it should be easy to add new tests to the original set to cover new problems and situations.

- The ReGreT system takes into account two elements to calculate the trust on an agent: the *direct trust* and the reputation. Similarly, the reputation value is a combination of different types of reputation (witness, neighbourhood, system, and default reputation). We have proposed a mechanism to aggregate all these elements to obtain a final trust value. We have argued that *direct trust* is more reliable than reputation and that witness reputation is better than neighbourhood reputation. Similarly, we have seen that neighbourhood reputation is better than system reputation. Using this ranking, the ReGreT system aggregates the different values to obtain a single and representative value.

  Although this method to aggregate the different elements is fine for certain scenarios, it is far from a general method that can be used in all possible scenarios. The set of experiments we have performed has shown that depending on the characteristics of the scenario the ranking we propose in this thesis is not the right one. In fact, we think that this general method doesn't exist.

For example, in a scenario where it is cheap and easy to obtain direct experiences it is better not to use reputation at all when the possibility that other agents give wrong information is high (for instance, because there is strong competition). Similarly, in those scenarios where the group policy is very important and the links between members are strong, neighbourhood and system reputation acquire a great relevance (even greater than direct experiences).

These are only two examples that show that the way the different elements of the system have to be combined cannot be static. Ideally, the agent should be clever enough to adapt the aggregation rules to the characteristics of the scenario even if these characteristics change along time.

This is not an easy task. The agent has to be able to extract the characteristics of the scenario analyzing the behaviour of the other agents. We think that further investigation in the use of social network analysis applied to agent societies is fundamental to achieve this objective.

- We have shown how the use of social network analysis can improve the capabilities of trust and reputation models. However, social network analysis techniques rely on the availability of accurate sociograms. We have to provide mechanisms that allow the agents to build and maintain these sociograms.

- We have to carry out more experiments oriented to analyze the use of social information as part of trust and reputation models. ReGreT is a first step toward the use of these techniques in trust and reputation models. More work in the same direction is necessary if we want to improve the efficiency of these models.

- Once the social dimension is introduced in trust and reputation systems and the agents start to take into account social relations, it becomes more and more important to consider not only which is the reputation of the other agents, but what can an agent do to get and maintain its own reputation. The agent needs mechanisms to analyze the impact of its actions in order to maintain its reputation in the society. This aspect of reputation needs further study.

- Further study on cognitive models is necessary. We are not claiming that cognitive models are a panacea. Probably the answer is in the use of hybrid models with cognitive as well as mathematical elements. However, up to now almost all the efforts have been directed to build mathematical models (check the list of models presented in chapter 2). It is time to analyze deeply what the cognitive approach can offer.

- Another area that requires further study is the use of trust and reputation models together with negotiation models, in other words, how trust and reputation can influence the negotiation strategies in order to improve the success of the negotiation process. In the *SuppWorld* framework examples that we have presented in chapter 7, agents use trust to decide if it is worth it or not to negotiate with a given partner. However, we have said anything about how these trust values can be used to modify the behaviour of the agent *during* the negotiation process. One

possibility could be the use of the trust value to decide how preservative or confident has to be the agent during the negotiation (that is, the degree of concession) or which has to be the acceptance/withdrawal point.

8.2.2 Multi-context systems

We have demonstrated that the use of multi-context systems for the design and implementation of autonomous agents is not only feasible but very convenient. However there are some aspects that have to be improved.

- We need to resolve the question of the semantics of the consuming conditions and time-outs in bridge rules. The inclusion of these two elements means that the model departs somewhat from first order predicate calculus, and so does not have a fully-defined semantics. As we have said in chapter 5, one possibility could be the use of linear logic, as a means of giving a semantics to consuming conditions, and various temporal logics as a means of giving a semantics to time-outs.
- It would be nice and very convenient to have a graphical tool for the rapid prototyping of declarative agents, with a library of different units and bridge rules which can be connected together and edited as required.
- In connexion with our work in the area of trust and reputation, we plan to use the multi-context approach for the specification and implementation of trust and reputation cognitive models. The characteristics of the multi-context approach makes this combination very promising and convenient.

Appendix A

Unit theories of the ReGreT module

A.1 ODB - Unit

$$contract(ID, Y, I, X^c, t) \wedge fulfillment(ID, Y, I, X^f, t') \rightarrow outcome(ID, Y, I, X^c, X^f, t')$$

A.2 DT - Unit

```
itm(10).

sum([],0).
sum([X|Xs],Res) :-
        sum(Xs,Res2),
        Res is X + Res2.

g(V,Res) :-
        itm(Itm),
        V =< Itm,
        !,
        Res is sin((3.1416 * V) / (2 * Itm)).
g(_,1).

dv([],[],_,_,[]).
dv([V|Vs],[T|Ts],SumT,DT,[R|Rs]) :-
        R is (T / SumT) * abs(V - DT),
        dv(Vs,Ts,SumT,DT,Rs).

dtVal([],[],_,[]).
dtVal([V|Vs],[T|Ts],SumT,[R|Rs]) :-
        R is (T / SumT) * V,
        dtVal(Vs,Ts,SumT,Rs).

dt(Y,Phi,DT,DTRL) :-
        findall(V,imp(_,Y,Phi,V,_),LV),
        findall(T,imp(_,Y,Phi,_,T),LT),
        sum(LT,SumT),
        dtVal(LV,LT,SumT,LR1),
        sum(LR1,DT),
        dv(LV,LT,SumT,DT,LR2),
        sum(LR2,Dv),
        length(LV,Length),
```

```
        g(Length,No),
        DTRL is No * (1 - Dv).
```

A.3 ICR - Unit

```
itm(10).

sum([],0).
sum([X|Xs],Res) :-
        sum(Xs,Res2),
        Res is X + Res2.

g(V,Res) :-
        itm(Itm),
        V =< Itm,
        !,
        Res is sin((3.1416 * V) / (2 * Itm)).
g(_,1).


dv([],_,[]).
dv([Rel|Rels],ICr,[R|Rs]) :-
        R is (Rel - ICr),
        dv(Rels,ICr,Rs).

iCr(W,Cr,CrRL) :-
        findall(R,wI(W,_,_,R),LR),
        sum(LR,SumR),
        length(LR,Length),
        Cr is SumR / Length,
        g(Length,No),
        dv(LR,Cr,LR2),
        sum(LR2,Dv),
        CrRL No * (1 - Dv).
```

A.4 SCR - Unit

```
- PROLOG:
check(W,Y,Coop,-1,false) :-
        (coop(W,Y,Coop) ; coop(Y,W,Coop)).
check(W,Y,-1,Comp,false) :-
        (comp(W,Y,Comp) ; comp(Y,W,Comp)).
check(_,_,-1,-1,true).

sCr(W,S,T,sCr(W,T,NUM_socialCr)) :-
        check(W,S,NUM_coop_ws,NUM_comp_ws,No_rel_ws),
        check(W,T,NUM_coop_wt,NUM_comp_wt,No_rel_wt),
        !,
        fuzzy_Inference_SCR(NUM_coop_ws,NUM_comp_ws,No_rel_ws,
                            NUM_coop_wt,NUM_comp_wt,No_rel_wt,NUM_socialCr).

- MILORD II:
Module fuzzy_Inference_SCR =
   Begin
      Import NUM_coop_ws, NUM_comp_ws, No_rel_ws
             NUM_coop_wt, NUM_comp_wt, No_rel_wt
      Export NUM_socialCr
      Deductive knowledge
      Dictionary:
         Predicates:
            NUM_coop_ws = Type: numeric
            NUM_comp_ws = Type: numeric
            No_rel_ws = Type: boolean
            NUM_coop_wt = Type: numeric
            NUM_comp_wt = Type: numeric
```

```
            No_rel_wt = Type: boolean
            NUM_socialCr = Type: numeric
                           Relation: needs_qualitative socialCr
            coop_ws = Type: (l "low" (0,0,0.2,0.4),
                             m "medium" (0.2,0.4,0.6,0.8),
                             h "high" (0.6,0.8,1,1))
                      Relation: needs_quantitative NUM_coop_ws
            comp_ws = Type: (l "low" (0,0,0.2,0.4),
                             m "medium" (0.2,0.4,0.6,0.8),
                             h "high" (0.6,0.8,1,1))
                      Relation: needs_quantitative NUM_comp_ws
            coop_wt = Type: (l "low" (0,0,0.2,0.4),
                             m "medium" (0.2,0.4,0.6,0.8),
                             h "high" (0.6,0.8,1,1))
                      Relation: needs_quantitative NUM_coop_wt
            comp_wt = Type: (l "low" (0,0,0.2,0.4),
                             m "medium" (0.2,0.4,0.6,0.8),
                             h "high" (0.6,0.8,1,1))
                      Relation: needs_quantitative NUM_comp_wt
            socialCr = Type: (vl "very low" (0,0,0.0625,0.1875),
                              l  "low" (0.0625,0.1875,0.3125,0.4375),
                              m  "moderate" (0.3125,0.4375,0.5625,0.6875),
                              h  "high" (0.5625,0.6875,0.8125,0.9375),
                              vh "very high" (0.8125,0.9375,1,1))

         Rules:
         R001 [If coop_ws is l
              then conclude socialCr is h] is true
         R002 [If coop_ws is m
              then conclude socialCr is vh] is true
         R003 [If coop_ws is h
              then conclude socialCr is vh] is true
         R004 [If comp_ws is l
              then conclude socialCr is m] is true
         R005 [If comp_ws is m
              then conclude socialCr is l] is true
         R006 [If comp_ws is h
              then conclude socialCr is vl] is true
         R007 [If coop_wt is l
              then conclude socialCr is m] is true
         R008 [If coop_wt is m
              then conclude socialCr is l] is true
         R009 [If coop_wt is h
              then conclude socialCr is vl] is true
         R010 [If comp_wt is l
              then conclude socialCr is m] is true
         R011 [If comp_wt is m
              then conclude socialCr is l] is true
         R012 [If comp_wt is h
              then conclude socialCr is vl] is true
         R013 [If No_rel_ws and No_rel_wt
              then conclude socialCr is h] is true
   End deductive
End
```

A.5 CR - Unit

```
wCr(W,WCr) :-
        iCr(W,ICr,ICrRL),
        sCr(W,SCr),
        WCr is ICrRL * ICr + (1-ICrRL) * SCr.
wCr(W,SCr) :-
        \+ iCr(W,_,_),
        sCr(W,SCr).
wCr(W,ICr) :-
        iCr(W,ICr,ICrRL),
```

```
        \+ sCr(W,_),
        ICrRL > 0.5.
wCr(W,0.5).
```

A.6 Witnesses - Unit

$$
\begin{aligned}
target(X) \wedge witness(W,X,\phi) \wedge trade(W,X) &\rightarrow tmp(X,\phi,\emptyset) \\
witness(W,X,\phi) \wedge tmp(X,\phi,S) &\rightarrow tmp(X,\phi,S \cup W) \\
tmp(X,\phi,S) \wedge (\nexists witness(W,X,\phi) \wedge trade(W,X)) | W \in S) &\rightarrow wSet(X,\phi,S)
\end{aligned}
$$

A.7 WRep - Unit

```
min(A,B,A) :-
        A < B.
min(_,B,B).

sum([],0).
sum([X|Xs],Res) :-
        sum(Xs,Res2),
        Res is X + Res2.

select(_,_,[],[],[]).
select(Y,Phi,[W|Ws],[Cr|Crs],[op(W,T,TRL)|Ops]) :-
        cR(W,Y,Cr),
        wI(W,Y,Phi,T,TRL),
        select(Y,Phi,Ws,Crs,Ops).
select(Y,Phi,[_|Ws],Crs,Ops) :-
        select(Y,Phi,Ws,Crs,Ops).

wRepVal(_,[],[],[],[]).
wRepVal(SumCr,[Cr|Crs],[op(_,T,TRL)|Ops],[R|Rs],[Rl|Rls]) :-
        R is (Cr / SumCr) * T,
        min(Cr,TRL,Min),
        Rl is (Cr / SumCr) * Min,
        wRepVal(SumCr,Crs,Ops,Rs,Rls).

wRep(Y,Phi,R,Rl) :-
        wSet(Y,Ws),
        select(Y,Phi,Ws,Crs,Ops),
        sum(Crs,SumCr),
        wRepVal(SumCr,Crs,Ops,Rs,Rls),
        !,
        sum(Rs,R),
        sum(Rls,Rl).
```

A.8 NiRep - Unit

```
- PROLOG:
rep(X,Phi,niRep(X,N,Phi,NUM_R,NUM_RL)) :-
        (coop(N,X,NUM_coop) ; coop(X,N,NUM_coop)),
        dt(N,Phi,NUM_dt,NUM_dtrl),
        fuzzy_Inference_R(NUM_dt,NUM_coop,NUM_R),
        fuzzy_Inference_RL(NUM_dtrl,NUM_coop,NUM_RL).

- MILORD:
Module fuzzy_Inference_R =
   Begin
      Import NUM_dt, NUM_coop
```

```
Export NUM_R
Deductive knowledge
Dictionary:
   Predicates:
      NUM_dt = Type: numeric
      NUM_coop = Type: numeric
      NUM_R = Type: numeric
              Relation: needs_qualitative R
      dt = Type: (vb "very bad" (-1,-1,-0.8,-0.7),
                  b  "bad" (-0.8,-0.7,-0.5,-0.4),
                  sb "slightly bad" (-0.5,-0.4,-0.2,-0.1),
                  n  "neutral" (-0.2,-0.1,0.1,0.2),
                  sg "slightly good" (0.1,0.2,0.4,0.5),
                  g  "good" (0.4,0.5,0.7,0.8),
                  vg "very good" (0.7,0.8,1,1))
           Relation: needs_quantitative NUM_dt
      coop = Type: (l "low" (0,0,0.2,0.4),
                    m "medium" (0.2,0.4,0.6,0.8),
                    h "high" (0.6,0.8,1,1))
             Relation: needs_quantitative NUM_coop
      R = Type: (vb "very bad" (-1,-1,-0.8,-0.7),
                 b  "bad" (-0.8,-0.7,-0.5,-0.4),
                 sb "slightly bad" (-0.5,-0.4,-0.2,-0.1),
                 n  "neutral" (-0.2,-0.1,0.1,0.2),
                 sg "slightly good" (0.1,0.2,0.4,0.5),
                 g  "good" (0.4,0.5,0.7,0.8),
                 vg "very good" (0.7,0.8,1,1))
   Rules:
   R001 [If dt is vb and coop is h
   then conclude R is vb] is true
   R002 [If dt is b and coop is h
   then conclude R is b] is true
   R003 [If dt is sb and coop is h
   then conclude R is sb] is true
   R004 [If dt is n and coop is h
   then conclude R is n] is true
   R005 [If dt is sg and coop is h
   then conclude R is sg] is true
   R006 [If dt is g and coop is h
   then conclude R is g] is true
   R007 [If dt is vg and coop is h
   then conclude R is vg] is true
   R008 [If dt is vb and coop is m
   then conclude R is vb] is almost-true
   R009 [If dt is b and coop is m
   then conclude R is b] is almost-true
   R010 [If dt is sb and coop is m
   then conclude R is sb] is almost-true
   R011 [If dt is n and coop is m
   then conclude R is n] is almost-true
   R012 [If dt is sg and coop is m
   then conclude R is sg] is almost-true
   R013 [If dt is g and coop is m
   then conclude R is g] is almost-true
   R014 [If dt is vg and coop is m
   then conclude R is vg] is almost-true
   R015 [If dt is vb and coop is l
   then conclude R is vb] is quite-true
   R016 [If dt is b and coop is l
   then conclude R is b] is quite-true
   R017 [If dt is sb and coop is l
   then conclude R is sb] is quite-true
   R018 [If dt is n and coop is l
   then conclude R is n] is quite-true
   R019 [If dt is sg and coop is l
   then conclude R is sg] is quite-true
   R020 [If dt is g and coop is l
   then conclude R is g] is quite-true
```

```
         R021 [If dt is vg and coop is l
         then conclude R is vg] is quite-true

   End deductive
End

Module fuzzy_Inference_RL =
   Begin
      Import NUM_dtrl, NUM_coop
      Export NUM_RL
      Deductive knowledge
      Dictionary:
         Predicates:
            NUM_dtrl = Type: numeric
            NUM_coop = Type: numeric
            NUM_RL = Type: numeric
                     Relation: needs_qualitative RL
            dtrl = Type: (vl "very low" (0,0,0.0625,0.1875),
                          l  "low" (0.0625,0.1875,0.3125,0.4375),
                          m  "moderate" (0.3125,0.4375,0.5625,0.6875),
                          h  "high" (0.5625,0.6875,0.8125,0.9375),
                          vh "very high" (0.8125,0.9375,1,1))
                   Relation: needs_quantitative NUM_dtrl
            coop = Type: (l "low" (0,0,0.2,0.4),
                          m "medium" (0.2,0.4,0.6,0.8),
                          h "high" (0.6,0.8,1,1))
                   Relation: needs_quantitative NUM_coop
            RL = Type: (vl "very low" (0,0,0.0625,0.1875),
                        l  "low" (0.0625,0.1875,0.3125,0.4375),
                        m  "moderate" (0.3125,0.4375,0.5625,0.6875),
                        h  "high" (0.5625,0.6875,0.8125,0.9375),
                        vh "very high" (0.8125,0.9375,1,1))
         Rules:
         R001 [If dtrl is vl and coop is l
         then conclude RL is vl] is true
         R002 [If dtrl is vl and coop is m
         then conclude RL is vl] is true
         R003 [If dtrl is vl and coop is h
         then conclude RL is vl] is true
         R004 [If dtrl is l and coop is l
         then conclude RL is vl] is true
         R005 [If dtrl is l and coop is m
         then conclude RL is vl] is true
         R006 [If dtrl is l and coop is h
         then conclude RL is l] is true
         R007 [If dtrl is m and coop is l
         then conclude RL is vl] is true
         R008 [If dtrl is m and coop is m
         then conclude RL is l] is true
         R009 [If dtrl is m and coop is h
         then conclude RL is m] is true
         R010 [If dtrl is h and coop is l
         then conclude RL is l] is true
         R011 [If dtrl is h and coop is m
         then conclude RL is m] is true
         R012 [If dtrl is h and coop is h
         then conclude RL is h] is true
         R013 [If dtrl is vh and coop is l
         then conclude RL is m] is true
         R014 [If dtrl is vh and coop is m
         then conclude RL is h] is true
         R015 [If dtrl is vh and coop is h
         then conclude RL is vh] is true
   End deductive
End
```

A.9 NRep - Unit

```
sum([],0).
sum([X|Xs],Res) :-
        sum(Xs,Res2),
        Res is X + Res2.

nRepVal([],[],_,[],[]).
nRepVal([R|Rs],[Rl|Rls],SumRl,[SR|SRs],[SRl|SRls]) :-
        SR is (Rl / SumRl) * R,
        SRl is (Rl / SumRl) * Rl,
        nRepVal(Rs,Rls,SumRl,SRs,SRls).

nRep(Y,Phi,NR,NRl) :-
        findall(R,niRep(_,Y,Phi,R,_),LR1),
        findall(Rl,niRep(_,Y,Phi,_,Rl),LRl1),
        sum(LRl1,SumRL),
        nRepVal(LR1,LRl1,SumRL,LR2,LRl2),
        sum(LR2,NR),
        sum(LRl2,NRl).
```

A.10 SRep - Unit

$$target(Y) \wedge inGroup(Y, G) \wedge groupRep(G, \phi, Rep) \rightarrow sRep(Y, \phi, Rep)$$

As part of the initial theory there are also a set of facts that relate a group and a behavioural aspect with the reputation value. These facts have the form:

$$groupRep(G, \phi, Rep)$$

where G is the group, ϕ the behavioural aspect and *Rep* the reputation value.

A.11 Rep - Unit

```
defaultRep(0,1).
defaultSRL(1).

getWRep(X,Phi,WRL,WR,WRL) :-
        wRep(X,Phi,WR,WRL).
getWRep(_,_,0,0,0).

getNRep(X,Phi,Ew,En,NR,NRL) :-
        nRep(X,Phi,NR,NRL),
        En is NRL * (1-Ew).
getNRep(_,_,_,0,0,0).

getSRep(X,Phi,Ew,En,Es,SR,SRL) :-
        defaultSRL(SRL),
        sRep(X,Phi,SR),
        Es is SRL * (1-Ew-En).
getSRep(_,_,_,_,0,0,0).

getDRep(_,_,Ew,En,Es,Ed,DR,DRL) :-
        defaultRep(DR,DRL),
        Ed is (1-Ew-En-Es).

reputation(X,Phi,R,Rl) :-
        getWRep(X,Phi,Ew,WR,WRl),
        getNRep(X,Phi,Ew,En,NR,NRl),
        getSRep(X,Phi,Ew,En,Es,SR,SRl),
        getDRep(X,Phi,Ew,En,Es,Ed,DR,DRl),
        R is (Ew * WR) + (En * NR) + (Es * SR) + (Ed * DR),
        Rl is (Ew * WRl) + (0.75 * En * NRl) + (0.5 * Es * SRl) +
              (0.25 * Ed * DRl).
```

A.12 Trust - Unit

```
getDT(Y,Phi,DT,DTRL) :-
        dt(Y,Phi,DT,DTRL).
getDT(_,_,0,0).

getRep(Y,Phi,R,RL) :-
        reputation(Y,Phi,R,RL).
getRep(_,_,0,0).

trust(Y,Phi,T,TRL) :-
        getDT(Y,Phi,DT,DTRL),
        getRep(Y,Phi,R,RL),
        T is (DTRL * DT) + ((1-DTRL) * R),
        TRL is (DTRL * DTRL) + ((1-DTRL) * RL).
```

Appendix B

Experiments' specification

B.1 Configuration file description

A *SuppWorld* configuration file has 9 sections. We will go through the parameters of each section one by one.

Section EXPERIMENT: This is the section for the general parameters of the experiment.

Parameter	Description
ExperimentID	An ID for the experiment.
numRounds	Number of rounds.
Sequence	This is an ordered list of the scenes that are executed each round. The first element is the number of scenes. Each scene is described by two or three characters depending on its type. The first character is the scene type. There are 4 possible types: • 'f' Input/output of products. This scene is both to simulate the entrance of rough material, the action of final consumers or the "pay day". • 'p' Production process. Transformation process that adds new value to a product. • 'c' Convention. • 'g' Market session. The second character indicates the home grid of the agents that will participate in that scene. The third is only used if the scene type is 'f'. In that case indicates if there is an entrance of rough material ('i'), the material has to be bought to simulate final consumers ('o') or we have to simulate a "pay day" ('s'). For example: "f2o" defines a final consumers simulation scene that will be performed with agents that have the home cell in grid number 2. "g1" defines a market session in grid number 1 (therefore the sellers will be the agent that have the home cell in grid number 1 and the buyers the agents that have the home cell in grid number 2).
EndSeq	Ordered list of scenes that are executed at the end of the experiment
Social_Level	Determines if the agents take into account the social relations (this is only applicable to cooperative and competitive relations, trade relations are always taken into account). There are two possible values: NON_SOCIAL (the agents do not take into account social relations) and SOCIAL_ALL (the agents take into account social relations). Social relations influence the fulfillment of contracts and the truthfulness of given information.
LogMode	Fixes the log mode. Depending on the value of this parameter, the *SuppWorld* framework monitors different aspects of the execution. There are three possibilities: SINGLE (the framework only monitors a selected subset of individuals), POPULUS (the framework monitors all the individuals) and NONE
IndToMonitor	This is the list of individuals that will be initialized. For each individual it is specified the ID and the type of trust model that will use after the initialization.
ResetRound	At this round, the individuals specified in the IndToMonitor parameter are initialized.

Section TRANSPORT: This section contains the list of different transport types available to sellers in order to send goods to the buyers.

Parameter	Description
type	The name of the transport.
ID	An ID for the transport.
speed	This is the number of ticks that the transport invests to move a set of goods from one cell to the adjacent.
cost	The cost (in terms of money) that has to move each unit of product from one cell to the adjacent.

Section PRODUCT PROFILES: Here you can specify the profile for each type of product in the supply chain.

Parameter	Description
type	The name of the product type.
profID	The ID of the product type.
inProduction	Number of product units of this type that are generated during an "Input of products" scene (the scene that simulates the entrance of rough material in the supply chain). This is the number of units that receives each agent.
inCost	The cost of each product unit if it is considered rough material (that is, if the agent obtains the product in a "Input of products" scene).
inLostCost	Money that the agent has to pay for each product unit that cannot be acquired during an "Input of products" scene (either because the agent does not have enough storage capacity, either because it has not enough money to buy the product unit).
productionCost	Cost to transform a unit of product in order to sell it in the next layer of the supply chain.
outProduction	Number of product units of this type that can be sold during an "Output of products" scene (when it simulates the action of final consumers). This is the number of units that can sell each agent.
outCost	The price of each product unit when it is sold in an "Output of products" scene simulating the action of final consumers.
Price_range	Allowed price range for each unit of the product when it is sold in a "market" scene.
Quantity_range	Minimum and maximum quantity of product that can be traded in a single negotiation process when the product is sold in a "market" scene.
Quality_range	Minimum and maximum quality that can have this product.
Transport	Transport types that can be used to deliver this product.

Section NEGO ENGINES: This section defines the parameters of the negotiation engines that use the agents to negotiate. Currently, the only strategy available is a tit-for-tat.

Parameter	Description
type	Negotiation engine type.
behaviour	This parameter modulates the degree of concession. As we have said, the negotiation engine follows a tit-for-tat strategy. Once the engine has decided how much it is going to concede (following the tit-for-tat strategy), this parameter is used to influence the degree of concession. There are five possible values $\{B, b, n, c, C\}$. 'B' and 'b' decrease the concession percentage dictated by the tit-for-tat strategy a 40% and a 20% respectively. 'n' do not modify the concession percentage. Finally, 'c' and 'C' increase the concession percentage a 20% and 40% respectively.
Price_Util	Weight of the issue 'Price' in the utility function.
Quantity_Util	Weight of the issue 'Quantity' in the utility function.
Quality_Util	Weight of the issue 'Quality' in the utility function.
TransType_Util	Weight of the issue 'TransType' in the utility function. It is mandatory that the summation of these four weights be equal to 1.

Section TRUST AND REPUTATION MODELS: This section specifies the parameters for the trust and reputation models. The set of parameters that specify a trust and/or reputation engine are dependent on the engine. Only the two first parameters (type, Model) are mandatory for all the engines. Here we will detail the parameters that specify the ReGreT system. As a convention, when we say that the parameter is a flag it means that a value of 1 activates the resource and a value of 0 deactivates it.

Parameter	Description
type	An ID for the set of parameters.
Model	The type of engine. In our examples will be always REGRET.
Model params	*These parameters allow to control which parts of the system are active.*
DirectTrust	Flag that activates the use of the direct trust module.
Reputation	Flag that activates the use of reputation.
WRep	Flag that activates the use of the witness reputation module.
Witness_itm_level	Fixes the *itm* value for the witness reputation module.
Credibility	Flag that activates the use of the credibility module.
infoCr	Flag that activates the credibility based on previous information.
socialCr	Flag that activates the credibility based on social network analysis.
DefaultCrValue	Default credibility value.
NRep	Flag that activates the use of the neighbourhood reputation module.
SRep	Flag that activates the use of the system reputation module.
DRep_Value	Default reputation value.
DRl_Value	Reliability for the default reputation value.
Expert systems	*These parameters specify the files that contain the information to instantiate the fuzzy systems of the ReGreT system.*
expertSocialCr	File with the rules to evaluate the credibility using social network analysis.
expertNRep	File with the specification for the neighbourhood reputation module (reputation values).
expertNRl	File with the specification for the neighbourhood reputation module (reliability values).
GrRelations	*This is the specification of the ontological dimension. The specification consists on a list of nodes and how they are related.*
node	Label that identifies the node and describes its associated behavioural aspect.
itm_level	Fixes the *itm* value. This parameter is used to calculate the reliability of a direct trust value when it is considered the number of outcomes that have been used to make the calculation.
child	Label identifying a child node that contributes to the calculation of the current node. If the current node is related with an atomic aspect of the behaviour, this label defines the issue of the contract associated to this aspect.
w	Relevance that has the child node value in the calculation of the parent node value. The summation of all the weights associated to a single parent node must be equal to 1.

Section INDIVIDUALS: This section describes the different types of individuals that will populate the supply chain.

Parameter	Description
type	Label that identifies this type of individual.
class	This parameter defines the base class of the agent depending on tis position in the supply chain. There are four possibilities: • 'PRODUCER' Agent that populates the first layer of the supply chain. • 'MANUFACTURER' Agent that inhabits the middle layers of the supply chain (that is, buy things in one market and sells the modified product in another market). • 'MANUFACTURER_F' Agent that populates the last layer of the supply chain and sells its products to a simulated final consumer. • 'CONSUMER' Agent that populates the last layer of the supply chain and receives a salary.
socID	Specifies which view of the society will be assigned to this type of agent (see section SOCIETY VIEWS).
FarmType	If the class is 'PRODUCER', this parameter fixes the quality of the product that the agents of this type will generate.
Cash	Initial amount of money.
PrIn	*This is a list of the products that needs this type of agent to produce the output product.*
PrType	An ID for the product.
productProfile	The profile for the product (it must be one of the profiles defined in section 'PRODUCT PROFILES').
MaxStock	Maximum capacity that has the agent to store this type of product (in number of units).

Stock	Initial stock of this kind of product at the beginning of the experiment.
StockQuality	Quality distribution of the initial stock.
PrOut	*This is the product that this type of agent produces and sells to the others. To produce one unit of this product the agent needs one unit of each product specified in the* PrIn *section*
PrType	An ID for the product.
productProfile	The profile for the product (it must be one of the profiles defined in section 'PRODUCT PROFILES').
MaxStock	Maximum capacity that has the agent to store this type of product (in number of units).
Stock	Initial stock of this kind of product at the beginning of the experiment.
StockQuality	Quality distribution of the initial stock.
NegoEngineStr	Negotiation engine that uses this type of agent. It must be one of the negotiation engines specified in the 'NEGO ENGINES' section.
TrustEngineStr	Trust engine that uses this type of agent. It must be one of the models specified in the 'TRUST AND REPUTATION MODELS' section.

Section MARKETS: This section describes the markets (grids) in the supply chain.

Parameter	Description
ID	An ID for the market.
dim	The size of the market.
sessionTime	Number of ticks that the market remains open during a session.
popTypes	List of agent types that can own a cell in this market. This parameter is used to generate a population randomly.
BehDstr	Fixes the behaviour of the agents that populate the market. The first element of the list represents the percentage of SAINTs, the second the percentage of GOOD agents and so on. The summation of the five elements must be equal to 100. Again, this is used to generate the agents behaviour of a population randomly but following a fixed distribution.
Grid specification	This section reproduces explicitly the distribution of agent types and behaviours over the market. Each position represents a physical cell on the market and the type and behaviour of its owner. For example, 'AB-G' represents an agent of type 'AB' with a GOOD behaviour. You can use '??' for the type and '?' for the behaviour to left the system to calculate randomly the type and behaviour according to the specifications in parameters 'popTypes' and 'NehDstr'.

Section CONVENTIONS: This section describes the convention scenes.

Parameter	Description
gridID	This parameter specifies the grid that hosts the agents that will participate in the convention.
sessionTime	The duration, in number of ticks, of the convention. During a tick, each agent that participates in the convention can make one single question to another agent.

Section SOCIETIES: This section specifies how are generated the sociograms that define the point of view of each agent about the society.

Parameter	Description
socID	The ID for this specific set of sociograms.
market	This specifies the set of agents that appear in the sociograms (in other words, the nodes of the sociograms). For instance, if 'market = grid1' means that the nodes of the sociograms are the agents that own a cell in 'grid1' (acting as sellers) and the agents that buy in that market (acting as buyers).
densityCoop	Density of the cooperation sociogram.
densityComp	Density of the competition sociogram.

B.2 Configuration files of the experiments

B.2.1 Scenario I

100% EVIL

```
// ************************************************************
// EXPERIMENT
// ************************************************************

ExperimentID   =  ExpID
numRounds      =  50
Sequence       =  [6,f1i,p1,c2,g1,p2,f2o]
EndSeq         =  [2,p2,f2o]
Social_Level   =  NON_SOCIAL
LogMode        =  SINGLE
IndToMonitor   =  [(2025|ALLNEGO),(2004|ALLNEGO),(2045|ALLNEGO),
                   (2054|ALLNEGO),(2032|ALLNEGO),(2026|ALLNEGO),
                   (2005|ALLNEGO),(2046|ALLNEGO),(2055|ALLNEGO),
                   (2033|ALLNEGO)]
ResetRound     =  30

// ************************************************************
// TRANSPORT
// ************************************************************

begin
    type  = BOAT
    ID    = 1
    speed = 1
    cost  = 1
-----
    type  = TRAIN
    ID    = 2
    speed = 2
    cost  = 2
-----
    type  = TRUCK
    ID    = 3
    speed = 3
    cost  = 3
-----
    type  = PLANE
    ID    = 4
    speed = 4
    cost  = 4
-----
    type  = SHUTTLE
    ID    = 5
    speed = 5
    cost  = 5
end

// ************************************************************
// PRODUCT PROFILES
// ************************************************************

begin
    type            =  ROUGH_PRODUCT
    profID          =  1
    inProduction    =  200
    inCost          =  10
    inLostCost      =  3
    productionCost  =  5
    outProduction   =  -1
    outCost         =  -1
    Price_range     =  [10,50]
    Quantity_range  =  [20,20]
    Quality_range   =  [1,5]
    Transport       =  [BOAT,TRAIN,TRUCK,PLANE,SHUTTLE]
-----
    type            =  MANUFACTURED_PRODUCT
    profID          =  2
    inProduction    =  -1
    inCost          =  -1
    inLostCost      =  0
    productionCost  =  15
    outProduction   =  -1
    outCost         =  20
    Price_range     =  [0,200]
    Quantity_range  =  [1,50]
    Quality_range   =  [1,5]
    Transport       =  [BOAT,TRAIN,TRUCK,PLANE,SHUTTLE]
end

// ************************************************************
// NEGO ENGINES
// ************************************************************
```

```
begin
    type            =  LINEAR
    behaviour       =  n
    Price_Util      =  0.5
    Quantity_Util   =  0
    Quality_Util    =  0.5
    TransType_Util  =  0
end

// ***********************************************************
// TRUST AND REPUTATION MODELS
// ***********************************************************
begin
    // ***************************************************************
    // ALWAYS NEGOTIATE
    // ***************************************************************
    type            =  ALLNEGO
    Model           =  REGRET

    // Model params
    DirectTrust         =  0
    Reputation          =  0
    WRep                =  0
    Witness_itm_level   =  5
    Credibility         =  0
    infoCr              =  0
    socialCr            =  0
    DefaultCrValue      =  0
    NRep                =  0
    SRep                =  0
    DRep_Value          =  0
    DRl_Value           =  0.3

    // Expert systems
    expertSocialCr      =  ../socialCr.fex
    expertNRep          =  ../NRep.fex
    expertNRl           =  ../NRl.fex

    // Ontological dimension
    begin
        node            =  offers_good_prices
        itm_level       =  3
        begin
            child               =  PRICE
            w                   =  1.0
        end
    -----
        node            =  offers_good_quality
        itm_level       =  3
        begin
            child               =  QUALITY
            w                   =  1.0
        end
    -----
        node            =  good_seller
        itm_level       =  -1
        begin
            child               =  price
            w                   =  0.5
        -----
            child               =  quality
            w                   =  0.5
        end
    end
-----
    // ***************************************************************
    // DT
    // ***************************************************************
    type            =  DT
    Model           =  REGRET

    // Model params
    DirectTrust         =  1
    Reputation          =  0
    WRep                =  0
    Witness_itm_level   =  5
    Credibility         =  0
    infoCr              =  0
    socialCr            =  0
    DefaultCrValue      =  0
    NRep                =  0
    SRep                =  0
    DRep_Value          =  0
    DRl_Value           =  0.5

    // Expert systems
    expertSocialCr      =  ../socialCr.fex
    expertNRep          =  ../NRep.fex
    expertNRl           =  ../NRl.fex
```

```
    // Ontological dimension
    begin
        node            =  offers_good_prices
        itm_level       =  3
        begin
            child               =  PRICE
            w                   =  1.0
        end
    -----
        node            =  offers_good_quality
        itm_level       =  3
        begin
            child               =  QUALITY
            w                   =  1.0
        end
    -----
        node            =  good_seller
        itm_level       =  -1
        begin
            child               =  price
            w                   =  0.5
        -----
            child               =  quality
            w                   =  0.5
        end
    end
-----
    // ************************************************************
    // DT + WR
    // ************************************************************
    type            =  DT+WR
    Model           =  REGRET

    // Model params
    DirectTrust         =  1
    Reputation          =  1
    WRep                =  1
    Witness_itm_level   =  5
    Credibility         =  0
    infoCr              =  0
    socialCr            =  0
    DefaultCrValue      =  0
    NRep                =  0
    SRep                =  0
    DRep_Value          =  0
    DRl_Value           =  0.3

    // Expert systems
    expertSocialCr      =  ../socialCr.fex
    expertNRep          =  ../NRep.fex
    expertNRl           =  ../NRl.fex

    // Ontological dimension
    begin
        node            =  offers_good_prices
        itm_level       =  3
        begin
            child               =  PRICE
            w                   =  1.0
        end
    -----
        node            =  offers_good_quality
        itm_level       =  3
        begin
            child               =  QUALITY
            w                   =  1.0
        end
    -----
        node            =  good_seller
        itm_level       =  -1
        begin
            child               =  price
            w                   =  0.5
        -----
            child               =  quality
            w                   =  0.5
        end
    end

// ************************************************************
// INDIVIDUALS
// ************************************************************
begin
    // **** PRODUCER: aA ***************************************
    type            =  aA
    class           =  PRODUCER
    socID           =  market1
    FarmType        =  4
    Cash            =  5000
```

```
    // PrIn
    begin
        PrType          =  a
        productProfile  =  ROUGH_PRODUCT
        MaxStock        =  1000
        Stock           =  0
        StockQuality    =  [0,0,0,0,0]
    end

    // PrOut
    PrType          =  A
    productProfile  =  ROUGH_PRODUCT
    MaxStock        =  1000
    Stock           =  0
    StockQuality    =  [0,0,0,0,0]

    // NegoEngine
    NegoEngineStr   =  LINEAR

    // ReputationEngine
    TrustEngineStr   =  NONE
-----
    // **** MANUFACTURER: AB ***********************************
    type            =  AB
    class           =  MANUFACTURER_F
    socID           =  market1
    FarmType        =  -1
    Cash            =  5000

    // PrIn
    begin
        PrType          =  A
        productProfile  =  ROUGH_PRODUCT
        MaxStock        =  100
        Stock           =  0
        StockQuality    =  [0,0,0,0,0]
    end

    // PrOut
    PrType          =  B
    productProfile  =  MANUFACTURED_PRODUCT
    MaxStock        =  100
    Stock           =  0
    StockQuality    =  [0,0,0,0,0]

    // NegoEngine
    NegoEngineStr   =  LINEAR

    // ReputationEngine
    TrustEngineStr   =  DT
end

// *****************************************************************
// MARKETS
// *****************************************************************

begin
    ID              =  grid1
    dim             =  [4,4]
    sessionTime     =  30
    popTypes        =  [aA]
    BehDstr         =  [0,0,0,0,100]
    begin
        aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?
    end
-----
    ID              =  grid2
    dim             =  [8,8]
    sessionTime     =  0
    popTypes        =  [AB]
    BehDstr         =  [100,0,0,0,0]
    begin
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?
    end
end

// *****************************************************************
// CONVENTIONS
// *****************************************************************
```

```
begin
    gridID          =  grid2
    sessionTime     =  1
end

// ************************************************************
// SOCIETY VIEWS
// ************************************************************

begin
    socID           =  market1
    market          =  grid1
    densityCoop     =  0.1
    densityComp     =  0.1
end
```

Taking the previous configuration file as a basis we compared four types of manufacturers. Each type of manufacturer was tested by modifying the following fields:

Manufacturer: NEGOTIATE ALWAYS

```
IndToMonitor   =  [(2025|ALLNEGO),(2004|ALLNEGO),(2045|ALLNEGO),
                   (2054|ALLNEGO),(2032|ALLNEGO),(2026|ALLNEGO),
                   (2005|ALLNEGO),(2046|ALLNEGO),(2055|ALLNEGO),
                   (2033|ALLNEGO)]
```

Manufacturer: DT + WR (LIERS)

```
IndToMonitor   =  [(2025|DT+WR),(2004|DT+WR),(2045|DT+WR),
                   (2054|DT+WR),(2032|DT+WR),(2026|DT+WR),
                   (2005|DT+WR),(2046|DT+WR),(2055|DT+WR),
                   (2033|DT+WR)]
BehDstr         =  [0,0,0,0,100]
```

Manufacturer: DT

```
IndToMonitor   =  [(2025|DT),(2004|DT),(2045|DT),
                   (2054|DT),(2032|DT),(2026|DT),
                   (2005|DT),(2046|DT),(2055|DT),
                   (2033|DT)]
```

Manufacturer: DT + WR

```
IndToMonitor   =  [(2025|DT+WR),(2004|DT+WR),(2045|DT+WR),
                   (2054|DT+WR),(2032|DT+WR),(2026|DT+WR),
                   (2005|DT+WR),(2046|DT+WR),(2055|DT+WR),
                   (2033|DT+WR)]
```

50% NEUTRAL - 50% EVIL

Same parameters of the experiment "*100% EVIL*" with the following difference in the *grid1* specification:

```
BehDstr         =  [0,0,50,0,50]
```

100% BAD

Same parameters of experiment "*100% EVIL*" with the following difference in the *grid1* specification:

```
BehDstr         =  [0,0,0,100,0]
```

50% NEUTRAK - 30% BAD - 20% EVIL

Same parameters of the experiment "*100% EVIL*" with the following difference in the *grid1* specification:

```
BehDstr         =  [0,0,50,30,20]
```

50% NEUTRAK - 30% BAD - 20% EVIL (LARGE SCENARIO)

Same parameters of the experiment "*100% EVIL*" changing the following fields:

```
// **************************************************************************************
// EXPERIMENT
// **************************************************************************************

numRounds      =  90
...
ResetRound     =  50

// **************************************************************************************
// GRIDS
// **************************************************************************************

begin
    ID              =  grid1
    dim             =  [8,8]
    sessionTime     =  60
    popTypes        =  [aA]
    BehDstr         =  [0,0,50,30,20]
    begin
        aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?
        aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?    aA-?
    end
-----
    ID              =  grid2
    dim             =  [16,16]
    sessionTime     =  0
    popTypes        =  [AB]
    BehDstr         =  [100,0,0,0,0]
    begin
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
        AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    AB-?    ...
    end
end

// **************************************************************************************
// CONVENTIONS
// **************************************************************************************

begin
    gridID          =  grid2
    sessionTime     =  20
end
```

B.2.2 Scenario II

For these experiments we use the configuration presented in B.2.1 with the following modifications:

```
Social_Level   =  SOCIAL_MAN
...
ResetRound     =  20
...

    // *************************************************************
    // DT + WR + SCr
    // *************************************************************
    type            =  DT+WR+SCr
    Model           =  REGRET

    // Model params
    DirectTrust         =  1
    Reputation          =  1
    WRep                =  1
    Witness_itm_level   =  5
    Credibility         =  1
    infoCr              =  0
```

```
    socialCr            =  1
    DefaultCrValue      =  0
    NRep                =  0
    SRep                =  0
    DRep_Value          =  0
    DRl_Value           =  0.3

    // Expert systems
    expertSocialCr      =  ../socialCr.fex
    expertNRep          =  ../NRep.fex
    expertNRl           =  ../NRl.fex

    // Ontological dimension
    begin
        node            =  offers_good_prices
        itm_level       =  3
        begin
            child               =  PRICE
            w                   =  1.0
        end
    -----
        node            =  offers_good_quality
        itm_level       =  3
        begin
            child               =  QUALITY
            w                   =  1.0
        end
    -----
        node            =  good_seller
        itm_level       =  -1
        begin
            child               =  price
            w                   =  0.5
        -----
            child               =  quality
            w                   =  0.5
        end
    end

...

// ************************************************************
// CONVENTIONS
// ************************************************************

begin
    gridID          =  grid2
    sessionTime     =  10
end
```

Comp Sociogram Density = 0.2

```
// ************************************************************
// SOCIETY VIEWS
// ************************************************************

begin
    socID           =  market1
    market          =  grid1
    densityCoop     =  0.0
    densityComp     =  0.2
end
```

Comp Sociogram Density = 0.5

```
// ************************************************************
// SOCIETY VIEWS
// ************************************************************

begin
    socID           =  market1
    market          =  grid1
    densityCoop     =  0.0
    densityComp     =  0.5
end
```

Comp Sociogram Density = 0.8

```
// ************************************************************
// SOCIETY VIEWS
// ************************************************************

begin
    socID           =  market1
    market          =  grid1
    densityCoop     =  0.0
    densityComp     =  0.8
end
```

Coop Sociogram Density = 0.2

```
// ************************************************************
// SOCIETY VIEWS
// ************************************************************

begin
    socID           =  market1
    market          =  grid1
    densityCoop     =  0.2
    densityComp     =  0.0
end
```

Coop Sociogram Density = 0.5

```
// ************************************************************
// SOCIETY VIEWS
// ************************************************************

begin
    socID           =  market1
    market          =  grid1
    densityCoop     =  0.5
    densityComp     =  0.0
end
```

Coop Sociogram Density = 0.8

```
// ************************************************************
// SOCIETY VIEWS
// ************************************************************

begin
    socID           =  market1
    market          =  grid1
    densityCoop     =  0.8
    densityComp     =  0.0
end
```

Bibliography

[Oxf, 2002] (2002). *The concise oxford disctionary of current english*. Oxford University Press.

[Abdul-Rahman and Hailes, 2000] Abdul-Rahman, A. and Hailes, S. (2000). Supporting trust in virtual communities. In *Proceedings of the Hawaii's International Conference on Systems Sciences, Maui, Hawaii*.

[Amazon, 2002] Amazon (2002). *Amazon Auctions*. http://auctions.amazon.com.

[Benerecetti et al., 1997] Benerecetti, M., Cimatti, A., Giunchiglia, E., Giunchiglia, F., and Serafini, L. (1997). Formal specification of beliefs in multi-agent systems. In Müller, J. P., Wooldridge, M. J., and Jennings, N. R., editors, *Intelligent Agents III*, pages 117–130. Springer Verlag, Berlin.

[Booch, 1994] Booch, G. (1994). *Object-oriented analysis and design with application*. Addison Wesley, Wokingham, UK.

[Brazier et al., 1995] Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N. R., and Treur, J. (1995). Formal specification of multi-agent systems. In *Proceedings of the 1st International Conference on Multi-Agent Systems*, pages 25–32.

[Bromley, 1993] Bromley, D. B. (1993). *Reputation, Image and Impression Management*. John Wiley & Sons.

[Buskens, 1998] Buskens, V. (1998). The social structure of trust. *Social Networks*, (20):265—298.

[Buskens, 1999] Buskens, V. (1999). *Social networks and trust*. PhD thesis, Utrecht University.

[Carbo et al., 2002a] Carbo, J., Molina, J., and Davila, J. (2002a). Comparing predictions of sporas vs. a fuzzy reputation agent system. In *3rd International Conference on Fuzzy Sets and Fuzzy Systems, Interlaken*, pages 147—153.

[Carbo et al., 2002b] Carbo, J., Molina, J., and Davila, J. (2002b). Trust management through fuzzy reputation. *Int. Journal in Cooperative Information Systems*, pages in–press.

[Carter et al., 2002] Carter, J., Bitting, E., and Ghorbani, A. (2002). Reputation formalization for an information-sharing multi-agent sytem. *Computational Intelligence*, 18(2):515—534.

[Castelfranchi et al., 1998] Castelfranchi, C., Conte, R., and Paolucci, M. (1998). Normative reputation and the cost of compliance. *Journal of Artificial Societies and Social Simulation (JASSS)*, 1(3).

[Castelfranchi and Falcone, 1998] Castelfranchi, C. and Falcone, R. (1998). Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS'98),Paris,France*, pages 72—79.

[Celentani et al., 1966] Celentani, M., Fudenberg, D., Levine, D., and Psendorfer, W. (1966). Maintaining a reputation against a long-lived opponent. *Econometrica*, 64(3):691—704.

[Cimatti and Serafini, 1995] Cimatti, A. and Serafini, L. (1995). Multi-agent reasoning with belief contexts: The approach and a case study. In Wooldridge, M. J. and Jennings, N. R., editors, *Intelligent Agents*, pages 62–73. Springer Verlag, Berlin.

[Dellarocas, 2002] Dellarocas, C. (2002). The digitalization of word-of-mouth: Promise and challenges of online reputation mechanisms. *in-press*.

[d'Inverno et al., 1998] d'Inverno, M., Kinny, D., Luck, M., and Wooldridge, M. (1998). A formal specification of dMARS. In Singh, M. P., Rao, A. S., and Wooldridge, M., editors, *Intelligent Agents IV*, pages 155–176. Springer Verlag, Berlin.

[Dubois and Prade, 1988] Dubois, D. and Prade, H. (1988). *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York, NY.

[eBay, 2002] eBay (2002). *eBay*. http://www.eBay.com.

[Esfandiari and Chandrasekharan, 2001] Esfandiari, B. and Chandrasekharan, S. (2001). On how agents make friends: Mechanisms for trust acquisition. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, Montreal, Canada*, pages 27—34.

[Faratin et al., 1997] Faratin, P., Sierra, C., and Jennings, N. (1997). Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, (24):159—182.

[Fisher, 1998] Fisher, M. (1998). Representing abstract agent architectures. In Müller, J. P., Singh, M. P., and Rao, A. S., editors, *Intelligent Agents V*, pages 227–242. Springer Verlag, Berlin.

[Gabbay, 1996] Gabbay, D. (1996). *Labelled Deductive Systems*. Oxford University Press, Oxford, UK.

[Giunchiglia, 1993] Giunchiglia, F. (1993). Contextual reasoning. In *Proceedings of the IJCAI Workshop on Using Knowledge in Context*.

[Giunchiglia and Serafini, 1994] Giunchiglia, F. and Serafini, L. (1994). Multilanguage hierarchical logics (or: How we can do without modal logics). *Artificial Intelligence*, 65:29–70.

[Glickman, 1999] Glickman, M. E. (1999). Parameter estimation in large dynamic paired comparison experiments. *Applied Statistics*, (48):377—394.

[Godo et al., 2002] Godo, L., Puyol-Gruart, J., and Sierra, C. (2002). Control techniques for complex reasoning: The case of Milord II. In Meyer, J.-J. C. and Treur, J., editors, *Agent-based Defeasible Control in Dynamic Environments*, volume 7 of *Handbook of Defeasible and Uncertainty Management Systems*, pages 65–97. Kluwer Academic Publishers. Invited paper. ISBN: 1-4020-0834-1.

[Grandison and Sloman, 2000] Grandison, T. and Sloman, M. (2000). A survey of trust in internet application.

[Hage and Harary, 1983] Hage, P. and Harary, F. (1983). *Structural Models in Anthropology*. Cambridge University Press.

[Hume, 1975] Hume, D. (1975). *A Treatise of Human Nature (1737)*. Oxford: Clarendon Press.

[Ingrand et al., 1992] Ingrand, F. F., Georgeff, M. P., and Rao, A. S. (1992). An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34–44.

[Jennings, 1995] Jennings, N. R. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75:195–240.

[Jennings, 1999] Jennings, N. R. (1999). Agent-based computing: Promise and perils. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1429–1436.

[Karlins and I.Abelson, 1970] Karlins, M. and I.Abelson, H. (1970). *Persuasion, how opinion and attitudes are changed*. Crosby Lockwood & Son.

[Lashkari et al., 1994] Lashkari, Y., Metral, M., and Maes, P. (1994). Collaborative interface agents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence,AAAI-Press*.

[Luhmann, 1979] Luhmann, N. (1979). *Trust and Power*. Chichester: Wiley.

[Marimon et al., 2000] Marimon, R., Nicolini, J., and Teles, P. (2000). Competition and reputation. In *Proceedings of the World Conference Econometric Society, Seattle*.

[Marsh, 1994] Marsh, S. (1994). *Formalising Trust as a Computational Concept*. PhD thesis, Department of Mathematics and Computer Science, University of Stirling.

[McKnight and Chervany, 2002] McKnight, D. H. and Chervany, N. L. (2002). Notions of reputation in multi-agent systems: A review.

[Meyer, 1998] Meyer, J. J. (1998). Agent languages and their relationship to other programming paradigms. In Müller, J. P., Singh, M. P., and Rao, A. S., editors, *Intelligent Agents V*, pages 309–316. Springer Verlag, Berlin.

[Mui et al., 2002] Mui, L., Mohtashemi, M., and Halberstadt, A. (2002). Notions of reputation in multi-agent systems: A review.

[Noriega and Sierra, 1996] Noriega, P. and Sierra, C. (1996). Towards layered dialogical agents. In Müller, J. P., Wooldridge, M. J., and Jennings, N. R., editors, *Intelligent Agents III*, pages 173–188, Berlin. Springer Verlag.

[OnSale, 2002] OnSale (2002). *OnSale*. http://www.onsale.com.

[Parsons and Giorgini, 1999] Parsons, S. and Giorgini, P. (1999). An approach to using degrees of belief in BDI agents. In Bouchon-Meunier, B., Yager, R. R., and Zadeh, L. A., editors, *Information, Uncertainty, Fusion*. Kluwer, Dordrecht.

[Parsons and Jennings, 1996] Parsons, S. and Jennings, N. R. (1996). Negotiation through argumentation—a preliminary report. In *Proceedings of the 2nd International Conference on Multi Agent Systems*, pages 267–274.

[Parsons et al., 1998] Parsons, S., Sierra, C., and Jennings, N. R. (1998). Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261—292.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[Plato, 1955] Plato (1955). *The Republic (370BC)*. Viking Press.

[Pujol et al., 2002] Pujol, J. M., Sangesa, R., and Delgado, J. (2002). Extracting reputation in multi-agent systems by means of social network topology. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems (AAMAS-02), Bologna, Italy*, pages 467—474.

[Pujol et al., 2003] Pujol, J. M., Sangesa, R., and Delgado, J. (2003). *Web Intelligence*, chapter A Ranking Algorithm Based on Graph Topology to Generate Reputation or Relevance. Springer Verlag.

[Puyol-Gruart, 1996] Puyol-Gruart, J. (1996). *MILORD II: A Language for Knowledge-Based Systems*, volume 1 of *Monografies del IIIA*. IIIA–CSIC. ISBN: 84–00–07499–8.

[Puyol-Gruart et al., 1998] Puyol-Gruart, J., Godo, L., and Sierra, C. (1998). Specialisation calculus and communication. *International Journal of Approximate Reasoning (IJAR)*, 18(1/2):107–130.

[Puyol-Gruart and Sierra, 1997] Puyol-Gruart, J. and Sierra, C. (1997). Milord II: a language description. *Mathware and Soft Computing*, 4(3):299–338.

[Sabater and Sierra, 2001] Sabater, J. and Sierra, C. (2001). Regret: A reputation model for gregarious societies. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, Montreal, Canada*, pages 61—69.

[Sabater and Sierra, 2002] Sabater, J. and Sierra, C. (2002). Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems (AAMAS-02), Bologna, Italy*, pages 475—482.

[Sabater et al., 1999] Sabater, J., Sierra, C., Parsons, S., and Jennings, N. R. (1999). Using multi-context systems to engineer executable agents. In Jennings, N. R. and Lespérance, Y., editors, *Intelligent Agents VI*, pages 277–294. Springer Verlag, Berlin.

[Schillo et al., 2000] Schillo, M., Funk, P., and Rovatsos, M. (2000). Using trust for detecting deceitful agents in artificial societites. *Applied Artificial Intelligence*, (Special Issue on Trust, Deception and Fraud in Agent Societies).

[Scott, 2000] Scott, J. (2000). *Social Network Analysis*. SAGE Publications.

[Sen and Sajja, 2002] Sen, S. and Sajja, N. (2002). Robustness of reputation-based trust: Booblean case. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems (AAMAS-02), Bologna, Italy*, pages 288—293.

[Shoham, 1993] Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60:51–92.

[Szyperski, 1998] Szyperski, C. (1998). *Component Software*. Addison Wesley, Wokingham, UK.

[Thomas, 1995] Thomas, S. R. (1995). The PLACA agent programming language. In Wooldridge, M. J. and Jennings, N. R., editors, *Intelligent Agents*, pages 355–370. Springer Verlag, Berlin.

[Treur, 1991] Treur, J. (1991). On the use of reflection principles in modelling complex reasoning. *International Journal of Intelligent Systems*, 6:277–294.

[Vila, 1995] Vila, L. (1995). *On temporal representation and reasoning in knowledge-based systems*. PhD thesis, Institut d'Investigació en Intelligencia Artficial.

[Weerasooriya et al., 1995] Weerasooriya, D., Rao, A., and Rammamohanarao, K. (1995). Design of a concurrent agent-oriented language. In Wooldridge, M. J. and Jennings, N. R., editors, *Intelligent Agents*, pages 386–402. Springer Verlag, Berlin.

[Wooldridge, 1996] Wooldridge, M. (1996). A knowledge-theoretic semantics for Concurrent MetateM. In Müller, J. P., Wooldridge, M. J., and Jennings, N. R., editors, *Intelligent Agents III*, pages 357–374. Springer Verlag, Berlin.

[Wooldridge, 1997] Wooldridge, M. (1997). Agent-based software engineering. *IEE Proceedings on Software Engineering*, 144:26–37.

[Wooldridge and Jennings, 1995] Wooldridge, M. J. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10:115–152.

[Yu and Singh, 2000] Yu, B. and Singh, M. P. (2000). A social mechanism of reputation management in electronic communities. In *Cooperative Information Agents (CIA), Boston, USA*, pages 154—165.

[Yu and Singh, 2001] Yu, B. and Singh, M. P. (2001). Towards a probabilistic model of distributed reputation management. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, Montreal, Canada*, pages 125—137.

[Yu and Singh, 2002a] Yu, B. and Singh, M. P. (2002a).

[Yu and Singh, 2002b] Yu, B. and Singh, M. P. (2002b). An evidential model of distributed reputation management. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems (AAMAS-02), Bologna, Italy*, pages 294—301.

[Zacharia, 1999] Zacharia, G. (1999). Collaborative reputation mechanisms for online communities. Master's thesis, Massachusetts Institute of Technology.

[Zadeh, 1975] Zadeh, L. (1975). Fuzzy logic and approximate reasoning. *Synthese*, 30:407–428.