



HAL
open science

Bridging the Gap between Social and Digital Worlds: System Modeling and Trust Evaluation

Naghm Alhadad

► **To cite this version:**

Naghm Alhadad. Bridging the Gap between Social and Digital Worlds: System Modeling and Trust Evaluation. Data Structures and Algorithms [cs.DS]. Université de Nantes, 2014. English. NNT : . tel-01112455

HAL Id: tel-01112455

<https://theses.hal.science/tel-01112455>

Submitted on 3 Feb 2015

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

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

UNIVERSITÉ DE NANTES
FACULTÉ DES SCIENCES ET DES TECHNIQUES

ÉCOLE DOCTORALE SCIENCES & TECHNOLOGIES
DE L'INFORMATION ET MATHÉMATIQUE - EDSTIM

Année 2014

Bridging the Gap between Social and Digital Worlds: System Modeling and Trust Evaluation

THÈSE DE DOCTORAT

Discipline : Informatique
Spécialité : Informatique

*Présentée
et soutenue publiquement par*

Nagham Alhadad

Le 20 juin 2014, devant le jury ci-dessous

Président	Pascal MOLLI, Professeur d'université, Université de Nantes
Rapporteurs	Marie-Christine ROUSSET, Professeur d'université, Université Joseph Fourier de Grenoble
	Bruno DEFUDE, Professeur d'université, TELECOM SudParis
Examineurs	Emmanuelle ANCEAUME, Chargée de Recherche, CNRS Rennes
	Christophe SIBERTIN-BLANC, Professeur d'université, Université Toulouse1-Capitole
	Patricia SERRANO-ALVARADO, Maître de conférence, Université de Nantes
	Yann BUSNEL, Maître de conférence, Université de Nantes

Directeur de thèse : Philippe LAMARRE, Professeur d'Université, INSA de Lyon

To my parents and my daughter.

ACKNOWLEDGMENTS

First and foremost, I would like to offer my gratitude to my supervisors Philippe Lamarre, Patricia Serrano-Alvarado and Yann Busnel for their belief, patience, support, guidance and immense knowledge. Without them this thesis would not have been completed or written. One simply could not wish for a better or friendlier supervisors. Philippe, Patricia, Yann: Thank you for being so wonderful persons.

I take this opportunity to thank the jury members for reading and reviewing this thesis and giving me useful feedbacks.

My special thanks goes out to Pascal Moli and Hala Skaf who have helped me settle down when I first arrived to France.

I would like to thank my dear friends Germaine, Rima, Liza, Aurélien, Diego, Mariana, Mariam, Jose, Prajol, Trija and Thomas Cerqueus for their support, encouragement and the nice moments we spent together. Also, my special thanks to all LINA members for their presence and support.

I would like to extend my special gratitude to my families in Syria and Jordan who have supported and encouraged me. I also thank my small family Khaled and Natalie for being in my life.

To all of you, thank you for being beside me.

CONTENTS

CONTENTS	v
1 INTRODUCTION	1
1.1 CONTEXT AND MOTIVATION	1
1.2 CONTRIBUTION	2
1.3 OUTLINE	3
1.4 PUBLICATIONS	4
2 A BACKGROUND ABOUT ENTERPRISE ARCHITECTURE	7
2.1 INTRODUCTION	9
2.2 ENTERPRISE ARCHITECTURE FRAMEWORK AND ENTERPRISE MODEL	10
2.2.1 The Open Group Architecture Framework (TOGAF) . . .	11
2.2.2 OBASHI	13
2.3 CONCLUSION	17
3 A BACKGROUND ABOUT TRUST	19
3.1 INTRODUCTION	21
3.2 TRUST CONCEPTS	21
3.2.1 Trust definition	21
3.2.2 Trust metrics	23
3.2.3 Local vs global trust	26
3.3 GRAPH-BASED APPROACHES FOR EVALUATING TRUST	26
3.3.1 Trust propagation	26
3.3.2 Dependent paths: a potential problem	29
3.4 CONCLUSION	31
4 SOCIOPATH	33
4.1 INTRODUCTION	35
4.2 SOCIOPATH METAMODEL	35
4.2.1 The social world	36
4.2.2 The digital world	36
4.2.3 The relations in SOCIOPATH	37
4.2.4 Example of a SOCIOPATH model: single PC	38
4.3 DEDUCED ACCESS AND CONTROL RELATIONS	39
4.4 SOCIOPATH DEFINITIONS	42
4.4.1 Activities and paths	43
4.4.2 Dependence	44
4.5 USE-CASE EXAMPLE: GOOGLEDOCS	46
4.6 CONCLUSION	50

5	SOCIOTRUST	51
5.1	INTRODUCTION	53
5.2	A SOCIOPATH MODEL AS A DIRECTED ACYCLIC GRAPH (DAG)	54
5.3	A PROBABILISTIC APPROACH TO INFER SYSTEM TRUST VALUE	55
5.3.1	Trust in a node for an activity	56
5.3.2	Trust in a path for an activity	56
5.3.3	Trust in a system for an activity	57
5.4	EXPERIMENTAL EVALUATIONS	59
5.4.1	Influence of the system architecture on the trust value	59
5.4.2	Influence of the path length and the number of paths on the trust value	60
5.4.3	Social evaluation: a real case	61
5.5	CONCLUSION	63
6	SUJECTIVETRUST	65
6.1	INTRODUCTION	67
6.2	INFERRING USER'S OPINION ON A SYSTEM USING SUBJECTIVE LOGIC	68
6.2.1	Opinion on a node for an activity	68
6.2.2	Opinion on a path for an activity	68
6.2.3	Opinion on a system for an activity	70
6.3	EXPERIMENTAL EVALUATION	73
6.3.1	Comparing the proposed methods	73
6.3.2	Studying the accuracy of the proposed methods	74
6.3.3	Social evaluation: a real case	76
6.4	CONCLUSION	78
7	CONCLUSION	79
	LIST OF FIGURES	82
	LIST OF TABLES	84
	BIBLIOGRAPHY	85
A	APPENDIXS	91
A.1	MATHEMATICAL PROOFS	93
A.1.1	Opinion on a path for an activity (mathematical proof)	93
A.1.2	Opinion on a system for an activity (mathematical proof)	99
A.1.3	Split Relations (mathematical proof)	106
A.1.4	Comparing Copy to mTNA	112
A.1.5	Comparing Copy to Split	114
A.2	EXPERIMENTAL EVALUATION	116
A.3	PROPOSED SURVEY	119

INTRODUCTION



1.1 CONTEXT AND MOTIVATION

In our daily life, we do social activities like chatting, buying, sending letters, working, visiting friends, *etc.* These activities are achieved in our society through physical, digital and human entities. For instance, if we want to *write and send a letter*, we might type it and print it, we might use a web application to indicate us the nearest mailbox then we might consult another web application to provide us the schedule of the public transport that will allow us to reach the mailbox.

Each entity in the previous system plays a role enabling us to achieve this activity. Our PC and printer should enable us to write the letter and print it. The public transport must allow us to reach the mailbox. The web applications should let us retrieve the necessary information about our travel. The persons who work in the postal service should send the letter in a reliable way.

Besides the explicit entities we identify, there are some implicit ones that play a role in the previous activity. For instance, the installed applications on our PC should work properly. Our Internet connection should allow us to access the web applications. The information that we retrieve from the web applications should be reliable. The providers of the physical and digital resources in the postal service should provide reliable resources.

The simple previous example illustrates that *under any activity we achieve every day, there is a whole system we **rely on** and we **trust**, maybe unconsciously, which enables us to perform our social activities.*

If we transpose our social life into a digital life, we find out the same structural aspects. Every day, digital activities like chatting, mailing, blogging, buying online and sharing data are achieved through systems composed of physical and digital resources *e.g.*, servers, software components, networks and PCs. These resources are provided and controlled by persons (individuals or legal entities) we depend on to execute these activities. The set of these entities and the different relations between them form a complex system for a specific activity. From this point of view, a digital system can be considered as a small society we rely on and we trust to perform our digital activities.

When users need to choose a system to perform a digital activity, they face a lot of available options. To choose a system, they evaluate it considering many criteria: functionality, ease of use, QoS, economical aspects, *etc.* Nowadays, trust is also a momentous aspect of choice.

From the previous, two main issues arise:

1. How to formalize the entities of a system and the relationships between them for a particular activity?
2. How to evaluate trust in a system as a whole for an activity, knowing that a system composes several entities, which can be persons, digital and physical resources?

These points embody the main focus of this thesis. We argue that studying trust in the separate entities that compose a system does not give a picture of how trustworthy a system is as a whole. The trust in a system depends on its architecture, more precisely, on the way the implicit and explicit entities the users depend on to do their activities, are organized. Thus, the challenge in evaluating trust in a system is firstly, *to model the system architecture for a specific activity*. Secondly, *to define the appropriate metrics to evaluate the user's trust in a modeled system for an activity*.

1.2 CONTRIBUTION

The contributions of this thesis are divided into two parts. The first part concerns the modeling of a system architecture, where we identify its components and the ways they are related. The second part uses the previous modeling to evaluate trust for an activity achieved through this system.

System modeling

Evaluating trust in a system for an activity requires a system modeling that makes the user aware of all the entities she depends on for achieving the activity.

We start by studying and analyzing existing frameworks that allow to model a system. We observe that this part of our work intersects with Enterprise Architecture domain that focuses on providing mechanisms to allow enterprises to model their systems. We find out that these models are not oriented to enable evaluating trust but predicting and developing the work of enterprises.

Inspired by this domain, we propose SOCIOPATH [ALB⁺12, ALB⁺11a, ALB⁺11b], a formal model to give a vision about a used system for an activity. This model formalizes the entities in a system for an activity and the relations between them.

Evaluating trust

To evaluate trust, we first, study the main notions about trust like trust definitions, models and metrics. Then we study the graph-based trust approaches that are mainly introduced in social networks. These approaches inspire us to present the model resulting from SOCIOPATH as a graph that represents a system for an activity then evaluate trust in this graph. However, the interpretation of the graph is different in our work from graph-based trust approaches in the literature. For us, a graph represents a system for a digital activity and not a social network. This assumption plays an important role in the operations we propose to evaluate trust and in the results we interpret.

We propose two new approaches to evaluate trust in a system for an activity, **SOCIOTRUST** and **SUBJECTIVETRUST**. **SOCIOTRUST** [ASABL13] uses probability theory and **SUBJECTIVETRUST** [ABSAL13] uses subjective logic.

1.3 OUTLINE

In the following, we detail the structure of this thesis :

Chapter 2: A background about enterprise architecture

This chapter gives the main concepts in the enterprise architecture modeling. It describes two main frameworks used in this domain, the points that intersects with our work and the necessity of proposing a simple model that is directly oriented to the global objective of this thesis which is evaluating trust in a system for an activity.

Chapter 3: A background about trust

In this chapter, we introduce the main concepts about trust like trust definitions, models, metrics, *etc.* We introduce an overview about subjective logic since it is used for evaluating trust. We focus on the works for evaluating trust in social networks.

Chapter 4: SocioPath: Modeling a system

In this chapter, we introduce our formal metamodel named **SOCIOPATH**. **SOCIOPATH** is based on notions coming from many fields, ranging from computer science to sociology. It is a generic metamodel that considers two worlds: the social world and the digital world. It allows to define models based on the first order logic and draw a graphical representation of these models that identify persons, hardware, software and the ways they are related. It also allows to answer the user on some main questions about her used system like, on which entities (either social or digital) she depends to achieve an activity? and what are the degrees of her dependencies on these entities?

Chapter 5: SocioTrust: Evaluating trust in a system for an activity using theory of probability

In this chapter, we show that by using **SOCIOPATH**, a system for an activity can be introduced as a simple graphical presentation. We use this presentation to evaluate trust.

Using probability theory, we propose an approach named **SOCIOTRUST** to evaluate trust in a system for an activity. Levels of trust are then defined for each node in the graph. By combining trust values, we are able to estimate two different granularities of trust, namely, *trust in a path* and *trust in a system*, both for an activity to be performed by a person. We conducted several experiments to analyze the impact of different characteristics of a system on the behavior of the obtained trust values. Experiments are conducted on both synthetic traces and real data sets that allowed us to validate the accuracy of our approach.

Chapter 6: SubjectiveTrust: Evaluating trust in a system for an activity using subjective logic

In this chapter, we use subjective logic to evaluate trust. This latter allows to express trust as subjective opinions with degrees of uncertainty, whereas in SOCIOTRUST, trust values are considered as values of probabilities. Hence, users cannot express their uncertainties.

We extend SOCIOTRUST to use subjective logic. The system model is also based on SOCIOPATH. Some experiments are conducted to evaluate the accuracy of this approach and to confront it with real users.

Chapter 7: Conclusion

This chapter concludes the thesis and provides future research lines.

1.4 PUBLICATIONS

This thesis is based on the following publications:

International conferences

- Nagham Alhadad, Yann Busnel, Patricia Serrano-Alvarado and Philippe Lamarre. Trust Evaluation of a System for an Activity with Subjective Logic. In *11th International Conference on Trust, Privacy, and Security in Digital Business (TrustBus2014)*, 12 pages, September, 2014, Munich, Germany. (submitted)
- Nagham Alhadad, Patricia Serrano-Alvarado, Yann Busnel and Philippe Lamarre. Trust Evaluation of a System for an Activity. In *10th International Conference on Trust, Privacy, and Security in Digital Business (TrustBus2013)*, 12 pages, August, 2013, Prague, Czech Republic.
- Nagham Alhadad, Philippe Lamarre, Yann Busnel, Patricia Serrano-Alvarado, Marco Biazzi and Christophe Sibertin-Blanc. SocioPath: Bridging the Gap between Digital and Social Worlds. In *23rd International Conference on Database and Expert Systems Applications (DEXA2012)*, short paper, 8 pages, September, 2012, Vienna, Austria.

National conferences and workshops

- Nagham Alhadad, Philippe Lamarre, Patricia Serrano-Alvarado and Yann Busnel. Trust Approach Based on User's Activities. In *Atelier Protection de la Vie Privée (APVP2012)*, 6 pages, Juin, 2012, Ile de Groix, France.
- Nagham Alhadad, Philippe Lamarre, Yann Busnel, Patricia Serrano-Alvarado and Marco Biazzi. SocioPath: In Whom You Trust?. In *Journées Bases de Données Avancées (BDA2011)*, short paper, 6 pages, October, 2011, Rabat, Morocco.
- Nagham Alhadad, Philippe Lamarre, Patricia Serrano-Alvarado, Yann Busnel and Marco Biazzi. SocioPath: In Whom You Trust?

In *Atelier Protection de la Vie Privée (APVP2011)*, 6 pages, Juin, 2011, Soreze, France.

Technical reports

- Nagham Alhadad, Yann Busnel, Patricia Serrano-Alvarado and Philippe Lamarre. Graph-Based Trust Model for Evaluating Trust Using Subjective Logic. *Research report* , 50 pages, October, 2013.
- Nagham Alhadad, Philippe Lamarre, Yann Busnel, Patricia Serrano-Alvarado, Marco Biazzi and Christophe Sibertin-Blanc. SocioPath: In Whom You Trust? *Research report* , 7 pages, September, 2011.

BACKGROUND ABOUT ENTERPRISE ARCHITECTURE

2

“If I had 8 hours to chop down a tree, I’d spend 6 hours sharpening my axe.”

-Abraham Lincoln.

CONTENTS

3.1	INTRODUCTION	21
3.2	TRUST CONCEPTS	21
3.2.1	Trust definition	21
3.2.2	Trust metrics	23
3.2.3	Local vs global trust	26
3.3	GRAPH-BASED APPROACHES FOR EVALUATING TRUST	26
3.3.1	Trust propagation	26
3.3.2	Dependent paths: a potential problem	29
3.4	CONCLUSION	31

IN order to focus on the first introduced question in the introduction: *How to formalize the entities of a system and the relationships between them for a particular activity?*, we study and analyze the Enterprise Architecture Framework (EAF) where models of enterprises can be created (*cf.* Section 2.2) including all the entities that build an enterprise, like human, digital and physical ones, and the relations between them. Since there are a variety of EAFs, we detail the main concepts in the most used frameworks in this domain: TOGAF (*cf.* Section 2.2.1), which is mostly used in huge enterprises and OBASHI (*cf.* Section 2.2.2), which is employed in relatively small enterprises. Then we show the necessity of proposing a simple model that is directly oriented to the global objective of this thesis which is evaluating trust in a system for an activity.

2.1 INTRODUCTION

In Enterprise Architecture (EA), the term “enterprise” does not mean only the information systems employed by an organization, it expresses also the whole complex, social-technical system [Gia10], including: people, information and technology.

A widely known definition of an EA is the definition given by Giachetti [Gia10]: “*An Enterprise Architecture is a rigorous description of the structure of an enterprise, which comprises enterprise components, the properties of those components, and the relationships between them*”.

The goal of this description is translating the business vision into models. To do that, analytical techniques are used to formalize an enterprise. This allows to produce models that describe the business processes, people organization, information resources, software applications and business capabilities within an enterprise. These models provide the keys that enable the enterprise evolution. Therefore, humans, technical resources, business information, enterprise goals, processes, the roles of each entity in an enterprise and the organizational structures should be included in this description.

Since the EA is a description that contains the different types of entities in an enterprise, it is divided into four layers [Hew06] as follows:

- The technology layer: it includes all the types of hardware like computers or networks, and software like operating systems.
- The application layer: it describes the structure of the applications that is built using the technology layer and the interaction between these applications. This structure should be created in a way that it can be reliable, available, manageable and reused.
- The data layer: it includes the information and the way they are stored, arranged and managed.
- The business layer: it describes the enterprise’s strategies, objectives, processes and activities.

Figure 2.1 illustrates the layers of the enterprise architecture.

Developing an EA is a complex task, it needs an expert that is called the *enterprise architect* who should own particular skills and knowledge, like the ability to see how parts interact with the whole, the knowledge of the business for which the enterprise architecture is being developed, interpersonal and leadership skills, communication skills, the ability to explain complex technical issues in a way that non-technical people may understand, the knowledge of Information Technology (IT) governance and operations, comprehensive knowledge of hardware, software, application, systems engineering and knowledge of financial modeling as it pertains to IT investment.

An enterprise architect builds a holistic view of the enterprise strategies, processes, information and resource assets [SR07]. She gets this knowledge and ensures that the IT environment used, which is supported by the enterprise architecture, meets the current and future company

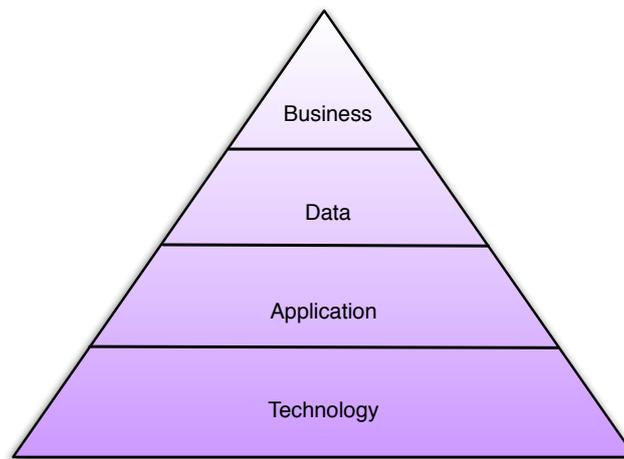


Figure 2.1 – Enterprise architecture layers.

needs in an efficient and adaptable manner. To do that, the enterprise architect uses a *framework* that provides some tools, which generate the enterprise *models*. In the next section, we explain the concepts of a framework and a model in an enterprise.

2.2 ENTERPRISE ARCHITECTURE FRAMEWORK AND ENTERPRISE MODEL

The Enterprise Architecture Framework (EAF) is a framework for an Enterprise Architecture which defines how to organize the structure and views associated with an Enterprise Architecture [The99].

An enterprise model is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise [MSF98].

The enterprise architecture is very complicated and large [Roh05]. To manage this complexity, the EAF provides methods and tools that allow to produce enterprise models. Many frameworks have appeared like the Zachman Framework in 1987 [Zac87], the Technical Architecture Framework for Information Management (TAFIM) in 1991 [Dep96], The Open Group Architecture Framework (TOGAF) in 1995 [Har07], The OBASHI Business & IT methodology and framework in 2001 [OBA] and The Rail Architecture Framework (TRAK) in 2003 [TRA03]. They all aim to analyze an enterprise by formalizing its entities and the relations between them using different analytical methods in order to produce enterprise models. To focus on this points, we choose to study two of the most used frameworks in this domain which follow the standard of the EA modeling, TOGAF and OBASHI.

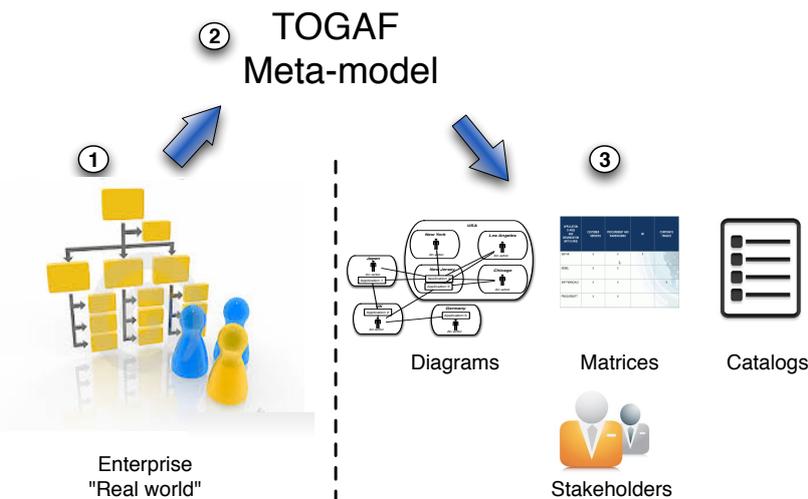


Figure 2.2 – Interactions between TOGAF metamodel, stakeholders and the models.

2.2.1 The Open Group Architecture Framework (TOGAF)

TOGAF is a framework for EA that provides an approach to design, implement and manage the enterprise information architecture [Har07]. It is a registered trademark of the Open Group in the United States and other countries [Jos09]. TOGAF defines a metamodel that allows to formalize an enterprise (*cf.* Section 2.2.1.1) and produce models (diagrams, catalogs and matrices) for the company stakeholders (*cf.* Section 2.2.1.2) as shown in Figure 2.2. Next sections introduce these concepts.

2.2.1.1 TOGAF metamodel

The TOGAF metamodel [Tog09b] allows to define a formal structure of the components within an architecture like an actor, a role, a data entity, an application and a business service. Besides the components, the metamodel defines the relationships between these components like an actor belongs to an organization unit or a role is assumed by an actor. Figure 2.3 shows a simplified version of the proposed metamodel in TOGAF.

TOGAF follows the standard of modeling at four layers: Technology, Application, Data and Business. The Technology level contains the *Technology Component*¹, which is a specific technology product. It can be either hardware or software like a specific type and version of a server or a specific type and version of an operating system. *Technology Component provides a platform for the application component and the business service.*

The Application level contains the *Application Component*, which is an application functionality like a purchase request processing application. An application component *is implemented on several technology components, accesses a data entity and supports a business service.* For example, the purchase request application would be implemented on several technology components, including hardware and software. It accesses some data like client information and supports the purchase order business service.

¹In this section, words in italic refer to keywords in the TOGAF metamodel shown in Figure 2.3.

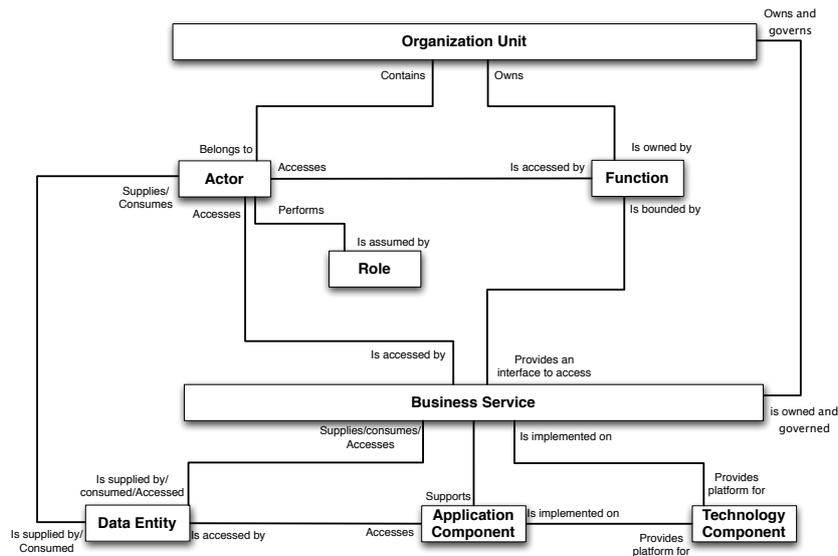


Figure 2.3 – A simplified version of TOGAF metamodel.

The Data level contains the *Data Entity*, which contains all the data of an enterprise. It is *supplied* or *consumed* by the *business service* or the *actor*. The Business level contains the following components:

- *Function*: a unit of business capability.
- *Business Service*: the services provided by the enterprise like a purchase order. A business service operates as a boundary for one or more functions. It *provides an interface to access a function*, it is *accessed by an actor* and *owned and governed by an organization unit*.
- *Organization unit*: an external or internal self-contained unit that has resources, responsibility and objectives. It *contains actors*, *owns functions* and *owns and governs business services*.
- *Actor*: a person or group of persons that interacts with the enterprise. An actor *accesses a business service* or a *function*, *consumes or supplies data entities* and *performs a role*.
- *Role*: the usual or expected task *assumed by an actor*.

The enterprise architects use the metamodel to formalize an enterprise architecture and provide some information represented in sub-models denoted catalogs, matrices and diagrams.

2.2.1.2 Catalog, matrix, and diagram models

The company stakeholders do not actually need to know the details of the metamodel or how the architecture has been built. They care about specific issues like “what is the most suitable team for a particular project?”, or “does the team have enough competencies to execute a given task?”.

In order to answer the needs of the stakeholders, the concepts of catalogs, matrices and diagrams have been defined in TOGAF.

APPLICATION (Y-AXIS) AND ORGANISATION UNIT (X-AXIS)	CUSTOMER SERVICES	PROCUREMENT AND WAREHOUSING	HR	CORPORATE FINANCE
SAP HR	X	X	X	
SIEBEL	X	X		
SAP FINANCIALS	X	X		X
PROCURESOFTE	X	X		

Figure 2.4 – System/Organization Matrix [Togoga].

Catalog: Catalogs are lists of components of a specific type, or of related types, that are used for governance or reference purposes [Togoga]. For example, the role catalog provides a list of all the actor roles within an enterprise. This catalog is a key input for the enterprise to identify the impact of organizational change management, to determine the company needs and to choose the qualified members for a specific task.

Matrix: Matrices are grids that show the relationships between two or more components from specific types [Togoga]. For instance, the System/Organization matrix represents the relationship between the application components and organizational units within the enterprise. This mapping helps the enterprise to define the application set that are used by a particular organization unit and facilitate the distribution of applications usage to the organization units that need it. Figure 2.4 shows an example of a System/Organization Matrix. For instance the organization “CORPORATE FINANCE” uses the application “SAP FINANCIALS”.

Diagrams: Diagrams are graphs that represent the components and the relationships between them in a rich and visual way to allow stakeholders to retrieve the required information easily [Togoga]. TOGAF defines a set of architecture diagrams to be created. For instance, the Application&User Location Diagram shows the locations of the actors and applications, and the actors who can access or use these applications as shown in Figure 2.5.

In the following, we present the layers, the relationships, the rules and the models of OBASHI another EAF.

2.2.2 OBASHI

The OBASHI framework provides a tool for capturing, illustrating and modeling the relationships of dependency and the dataflows between business and IT environment in a business context. The name OBASHI is a licensed trademark of OBASHI Ltd [OBA].

Contrarily to TOGAF, OBASHI does not have a specific metamodel to

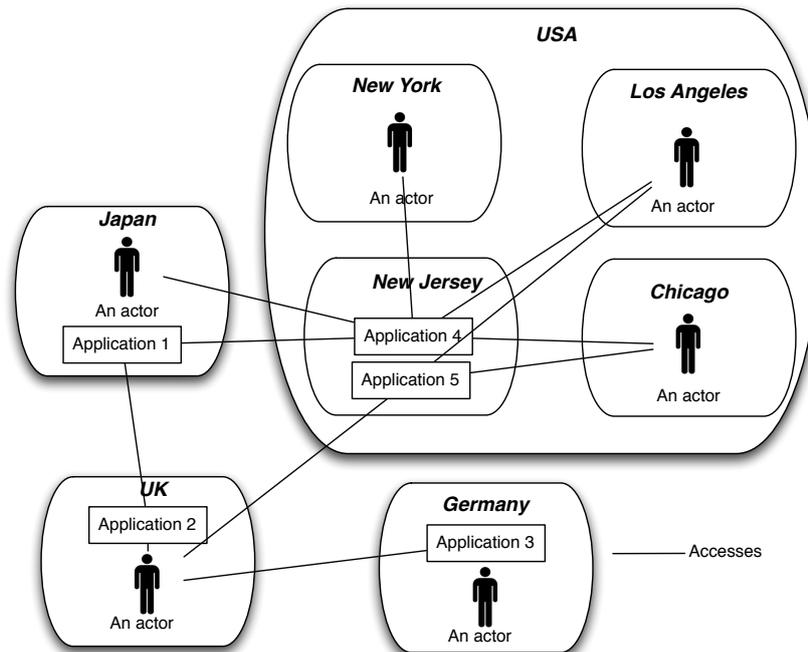


Figure 2.5 – *Application&User Location diagram.*

formalize the enterprise components. Instead, OBASHI proposes an identified classification for the components which should be located in the layer that corresponds to their type. OBASHI has six layers: **O**wnership, **B**usiness processes, **A**pplications, **S**ystems, **H**ardware, and **I**nfrastructure (*cf.* Section 2.2.2.1). The OBASHI relationships (*cf.* Section 2.2.2.2) describe the relations between the components, which follow the OBASHI rules (*cf.* Section 2.2.2.3). This model allows to create the business and IT diagram (B&IT) and the dataflow (*cf.* Section 2.2.2.4), which are the main output of the OBASHI tool that helps the enterprise to develop its work and understand its needs.

2.2.2.1 OBASHI Layers

OBASHI extends the EA modeling from four layers to six layers [OBA]. It divides the technology layer of EA into three layers: infrastructure, hardware and software. The business layer is divided into a business process layer and an ownership layer. The data layer is included implicitly in the business process layer.

- **Infrastructure:** the network infrastructure into which the hardware is connected, *e.g.*, Switches, Routers and Hubs.
- **Hardware:** the computer hardware on which the operating systems run, *e.g.*, Servers, Laptops, Tablet and PCs.
- **System:** the operating system on which the applications run, *e.g.*, Unix, Linux, Windows XP and Vista.
- **Application:** software application, *e.g.*, Excel, Oracle, *etc.*

- Business Process: the processes or functions that are used by the owners, *e.g.*, Monthly Balance, Sales Transactions or Capture Budgeting.
- Ownership: the person or group of persons that owns or governs business processes that exist in the Business Process Layer, *e.g.*, Planning Manager or Purchasing Officer.

The components are located in the layer that corresponds to their type. Then relationships between them are defined as we show in the next section.

2.2.2.2 OBASHI relationships

Six types of relationships are defined in OBASHI:

- Connection: a relation that represents a physical connection between two components.
- Dependency: a relation that shows when a component depends on another in order to function normally.
- Spatial: an implied relationship that exists between components placed above or below each other in OBASHI layers.
- Set: explicit logical group of components regardless of their position in the OBASHI layers.
- Layer: components that belong to the same layer.
- Sequential: list of components, which have a sequence of dependency or connection relationship.

The relationships between the components follow the OBASHI rules, as introduced below.

2.2.2.3 OBASHI rules

The following rules govern the implicit and explicit relationships between the enterprise components in the OBASHI Framework.

- The connection relationship is a bi-directional relationship.
- A component can be connected to one or more components.
- The connection relation exists between components in the same layer or between components in two adjacent layers.
- The components within the same layer have an implicit relationship between each other, which is the layer relation.
- The dependency relation is a uni-directional relationship *i.e.*, a component *X* may be dependent on a component *Y*, but *Y* might not be dependent on *X*.

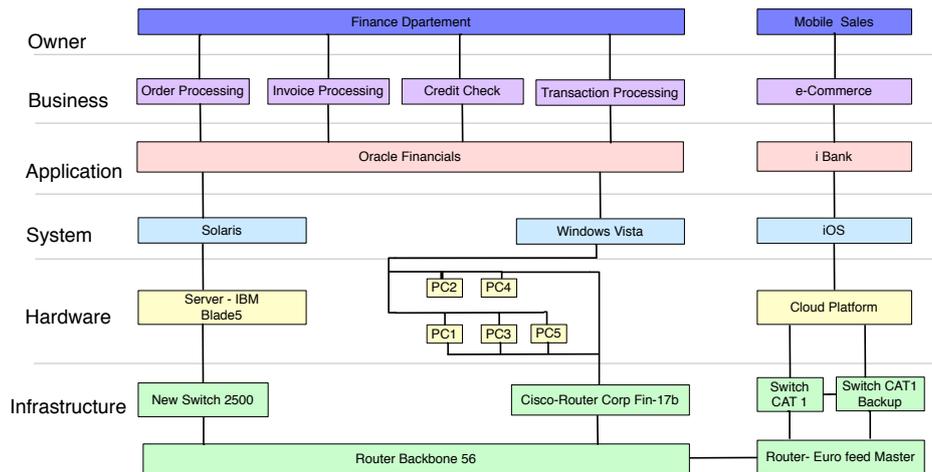


Figure 2.6 – OBASHI B&IT diagram [OBA].

- The sequential relationship comprises at least two connected components.
- A component may have one or more instances within a layer.

After situating the components in the layers, defining the relationships between them and applying the OBASHI rules, we obtain the B&IT diagram and the dataflows which are considered as the main output of the OBASHI tool.

2.2.2.4 OBASHI B&IT and dataflow

The B&IT diagram is a graphical representation of the enterprise components and the relationships between them, which provides a visual map of how the business works and bridges the gap between the business and the IT. This view helps to manipulate and analyze the enterprise information within its business context. Figure 2.6 shows an example of an OBASHI B&IT diagram where each component is situated in the layer that corresponds to it (*cf.* Section 2.2.2.1) and the relations between these components are applied (*cf.* Section 2.2.2.2) with respect to OBASHI rules (*cf.* Section 2.2.2.3).

The business resources and IT assets in OBASHI approach are considered as the path through which the data can flow [OBA10]. Based on that, OBASHI provides a graphical representation about the data flows across the organization. This is called OBASHI dataflow, which is achieved by following the sequence of the connection and dependency relations in the B&IT diagram.

The dataflow helps to track dataflows between people, process and technology, which provide an intuitive and easily understandable diagram that facilitates the management, the communication and the decision making. Figure 2.7 shows an example of a dataflow view extracted from Figure 2.6.

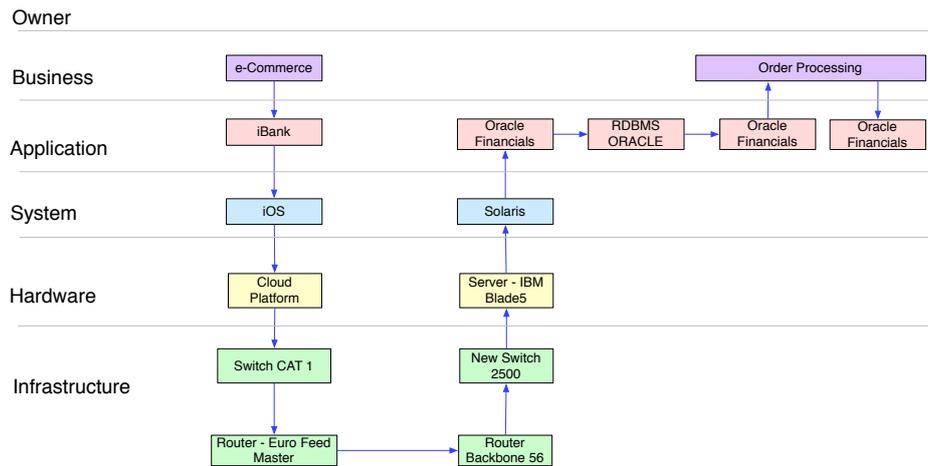


Figure 2.7 – OBASHI dataflow.

2.3 CONCLUSION

In this chapter, we gave an overall view about the main aspects in EA. We presented two enterprise architecture frameworks, TOGAF, which is a very powerful framework that is used in huge enterprises, and OBASHI, which is relatively simple and easy-used (much more employed in small enterprises). With this general view, we have focused on the first part of the scope of this thesis.

TOGAF is a very rich and powerful framework. It proposes a detailed complex metamodel that allows to retrieve information about an enterprise which useful for developing the enterprise work. TOGAF produces a set of graphs represented by *diagrams*. None of these diagrams represents directly an activity achieved through a system for a given user. Besides that, obtaining these diagrams goes under a complex procedure of modeling that needs an expert to do it. This complexity and the high cost of TOGAF leads us to exclude this framework and study a simpler one, OBASHI.

Contrarily to TOGAF, OBASHI does not have a specific metamodel to formalize the architecture. Instead, OBASHI proposes an identified classification for the components which should be located in the layer that corresponds to their type. For instance, a PC is situated in the layer Hardware, an operating system is situated in the layer System. The relationships between these components follow some defined rules that control them. Despite the simplicity of using OBASHI, the introduced output does not answer our needs. The B&IT Diagram and the data flow present a dependency graph that allows to find the sequences of the dependencies relations between the entities in an enterprise. In our work, the resulted model should represent an activity achieved through a system by a given user, more precisely, the model should contain the entities this user depends on to perform an activity and not the flow of dependencies between entities in a system.

Generally, in EA, modeling covers all the complexity of an enterprise including the financial aspects, processes, projects, *etc.* Our work con-

verges with the EA works in the general structure as we both aim to model an architecture to retrieve some information about the used system.

However, in our work, we look for a solution to describe a complex system, potentially based on several enterprises. Through this description, we aim to retrieve the information that concerns an activity achieved by a given person through a system, including the entities this person depends on for this activity which can be human, physical or digital ones and can belong to different organizations. Our objective focuses on modeling a system from the point of view of the user who achieves an activity and not from the point of view of the enterprise owners who aim to make global analysis that improves performance as in EA. Furthermore, the social aspects in EA has been limited to the business or owner level. A human in EA has been considered as an actor that ensures business continuity. Whereas, in our work, the analysis of information are centralized on the participants to a system including the potential trust relations between them. These main differences in the objective and the viewpoints impose different notions in the modeling, different degrees of complexity and different approaches to follow but all inspired from the concepts in the EA works which are presented in this chapter as we will show in Chapter 4.

In the second part of this thesis, we use the system modeling to evaluate the user trust in an architecture. Next chapter presents a background about the notion of trust.

A BACKGROUND ABOUT TRUST

3

“Trust is a social good to be protected just as much as the air we breathe or the water we drink. When it is damaged, the continuity of the whole suffers; and when it is destroyed, societies falter and collapse.”

-Sissela Bok.

CONTENTS

4.1	INTRODUCTION	35
4.2	SOCIOPATH METAMODEL	35
4.2.1	The social world	36
4.2.2	The digital world	36
4.2.3	The relations in SOCIOPATH	37
4.2.4	Example of a SOCIOPATH model: single PC	38
4.3	DEDUCED ACCESS AND CONTROL RELATIONS	39
4.4	SOCIOPATH DEFINITIONS	42
4.4.1	Activities and paths	43
4.4.2	Dependence	44
4.5	USE-CASE EXAMPLE: GOOGLEDOCS	46
4.6	CONCLUSION	50

IN order to focus on the second introduced question in the introduction: *How to evaluate trust in a system as a whole for an activity, knowing that a system composes several entities, which can be persons, digital and physical resources?*, we study the main concepts about trust like definitions, metrics and types and we analyze the proposed approaches for evaluating trust. We proceed from a broad view of trust in different disciplines reaching the computer science in Section 3.2.1, then we present some main concepts about trust in Sections 3.2.2 , 3.2.3. We finally focus on trust studies which are based on a graph in Section 3.3 since these works evaluate trust depending on a model (graph) that represents a used system.

3.1 INTRODUCTION

Trust plays an important role in our daily life. We build trust relationships every day without being aware of that. We drive our cars because we trust some car companies to produce safe and reliable cars. We work in a team because we trust our colleagues, directors and institutions. We live in our houses because we trust the architects and the workers who build it. Brief, our society cannot exist or survive without trust.

Since the society relies that heavily on trust among its members [Coo01, Uslo2], it has been studied in many sciences ranging from sociology, psychology, philosophy to economics and computer science.

In computer science, a system can be considered as a small society in which each entity is an individual. When users need to choose a system to perform a digital activity, they are faced with a lot of available options. They evaluate it considering many criteria: functionality, ease of use, QoS, economical aspects, *etc.* Recently, *trust* emerged as a momentous aspect of choice [JIB07, Mar94]. In the trust management community [MFGL12, Vil05, JIB07, ZDB11, YHo7], two main issues arise: (i) *How to define the trust in an entity, knowing that entities can be persons, digital and physical resources?* Defining trust in each type of entity is naturally different but mainly depends on *subjective* and *objective* properties [YHo7]. The second issue is, (ii) *How to evaluate such value of trust in a system under a particular context?* This point embodies the main focus of our study. Since our objective is evaluating trust in a system for an activity, some main concepts about trust are studied in Section 3.2: trust definitions, trust metrics and trust types. In Section 3.3, we focus on the graph-based trust approaches for evaluating trust which are studied in the literature.

3.2 TRUST CONCEPTS

In this section, we present some definitions and models of trust in different domains and the used metrics for evaluating trust. We also present the notion of global and local trust.

3.2.1 Trust definition

Researchers define trust according to their specific world view. Psychologists define trust as “*cognition about the trustee*” [RHZ85]. For some philosophers, trust is “*the acceptance of the risk of being betrayed*” [McLo6], while economists define trust as “*an economic-choice mechanism that reduces the cost of transactions, enables new forms of cooperation and generally furthers business activities, employment and prosperity*” [Fuk96].

The multiple definitions of trust have prompted some researchers to develop a general model of trust based on the different sciences. The study of McKnight *et al.* [MCC98] about trust in social sciences is frequently cited by computer scientists because it presents a broader view about trust models and trust properties. The components that compose his trust model shown in Figure 3.1 are the following:

Disposition to trust (mainly from psychology): means a general propensity to trust others.

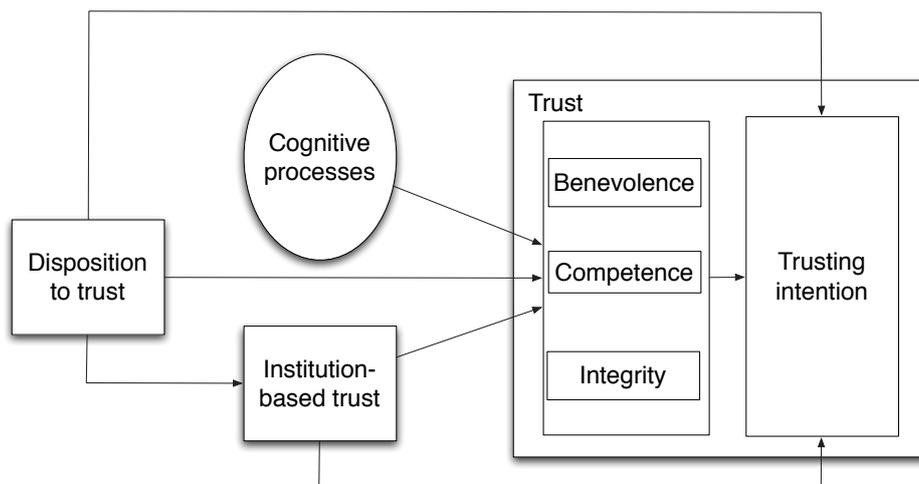


Figure 3.1 – McKnight's trust model [MCC98].

Institution-based trust (mainly from sociology): refers to an individual's perception of the institutional environment.

Cognitive processes (mainly from psychology): means the presuppose of the availability of knowledge like reputation.

Trusting beliefs (mainly from social psychology): means that the trustor realizes if the trustee has attributes that are beneficial to her, like the ability of the trustee to do what the trustor needs (*Competence*), the trustee caring and motivation to act in the trustor's interests (*Benevolence*) and the trustee honesty and promise keeping (*Integrity*).

Trusting intention (mainly from social psychology): means the trustor is securely willing to depend, or intends to depend, on the trustee.

Mayer *et al.* [RCMS95] also developed a general model of trust considering many concepts that come from different sciences. Figure 3.2 illustrates their proposed model. This model converges with the previous one in most of the considered aspects about trust like the *integrity* and the *benevolence* of the trustee. The concepts of *ability* and *propensity* corresponds respectively to the concepts of *competence* and *disposition to trust* in McKnight's model. Mayer added a new concept to his model, he considers that *risk* should be a factor that affects trust.

The previous models are generic and comprehensive but they are complex and highly theoretical, besides the difficulties in implementations. Yet, computer scientists often adopt trust definitions in social sciences because they consider them: simple, real and implementable. Golbeck [Golo5] defines trust as "Alice trusts Bob if she commits an action based on a belief that Bob's future actions will lead to a good outcome", which is adopted from sociology [Deu62]. Jøsang *et al.* [JIB07] define trust as "the subjective probability by which an individual expects that another individual performs a given action on which its welfare depends". This definition is adopted from the work of the sociologist Gambetta [Gam00].

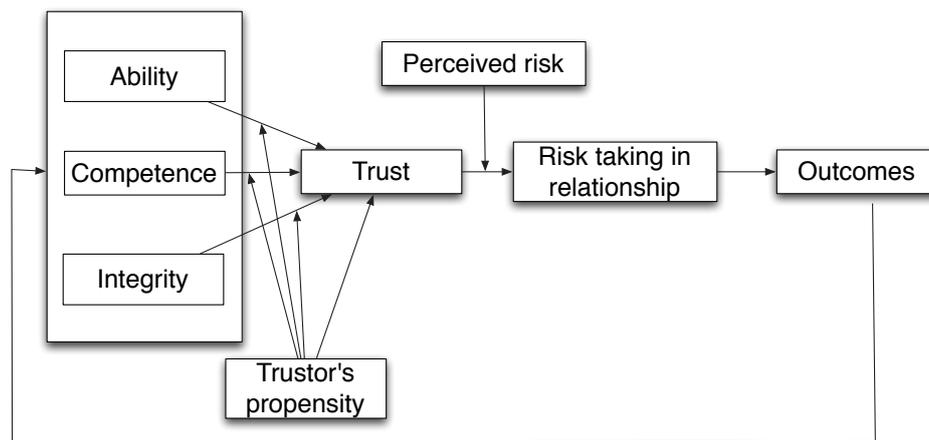


Figure 3.2 – Mayer's trust model [RCMS95].

3.2.2 Trust metrics

Trust metrics can be classified according to the used scale of trust values. Several types of trust metrics have been proposed. To simplify their description, we classify them into two main categories; discrete metrics where a trust value belongs to a defined discrete interval like $\{1, 2, 3, 4, 5\}$ and continuous metrics where a trust value belongs to a defined continuous interval like $[0, 1]$.

3.2.2.1 Discrete metrics

The simplest way to define a discrete metric is by using binary values that express either *trust* or *distrust* like [CH97, GH06, KF]02]. Other works discuss that trust should have a degree of strength [ARH00, CNS03, Man98, VC03]. Thus, it should be represented as a range of discrete values like eBay¹, where trust values are +1 to give a positive opinion, 0 for a neutral opinion, or -1 for a negative opinion. In Epinions², trust values are integer values in the range $\{1, 2, 3, 4, 5\}$. Abdul-Rahman *et al.* [ARH00] propose to give semantic meaning to the values. In their proposition, users vote *very trustworthy*, *trustworthy*, *untrustworthy* or *very untrustworthy*.

Users are often better able to provide a trust value in the form of discrete statements, than in the form of continuous measures. A system that allows trust to be expressed in the form of a discrete statement provides better usability than in the form of a probability value. This is because the meaning of discrete statements comes to mind immediately, whereas probability values require more cognitive effort to be interpreted [Jø07].

The operations that are applied to the discrete metrics for evaluating a trust value are formed by considering relatively simple or basic methods of computation like the multiplication, the minimum, the maximum, the average of several trust values [SKJ⁺00] or the weighted average. The weight can be determined by several factors like the credibility of the person who evaluates the trustee [GH06].

¹www.ebay.com

²www.Epinions.com

Although that systems that allow discrete metrics provide better usability than continuous metrics because users can express trust easily, researchers consider that discrete measures do not easily lend themselves to sound computational principles [JIB07, Jøso7, TA12]. Thus continuous metrics have been proposed.

3.2.2.2 Continuous metrics

The continuous range $[0, 1]$ of trust values has been proposed to cover the limitations of the computing methods used for discrete metrics. Trust value in the continuous range represents the probability of how much a trustee will perform actions in the way the trustor expects. Trust studies in computer science are often based on works made in sociology or psychology (*cf.* Section 3.2.1). If we turn back to works on trust in sociology, some of them define trust as a value of probability.

The sociologist Gambetta [Gamoo] says *“when we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him.”*

Computer scientists did not hesitate to capture this notion and interpret it as mathematical formalism that can be implemented.

In probabilistic metrics, trust values are usually between 0 and 1. In [LW10], authors compute a trust value towards composite services. The trust value in each service is considered as a probability by which a service performs an expected action, then the rules of independent probability and conditional probability are applied to compute trust in composite service. The advantage of probabilistic models is that the rich body of probabilistic methods can be directly applied. This provides a great variety of possible derivation methods, from simple models based on probability calculus to models using advanced statistical methods.

Despite the framework that is provided by probabilistic metrics to introduce a meaningful computation for trust, researchers argue that the probability value is not totally realistic in introducing trust because users cannot express their ignorance or their degree of uncertainty about a subject. In simple words they cannot say *I do not know* or *I am not sure*. Users hardly can determine an absolute certainty opinion on a proposition. This idea led researchers to look for mathematical formalisms to express uncertainty.

Subjective logic [Jøso1] proposes a solution to this problem. It is a probabilistic logic where the sum of probabilities over all possible outcomes not necessarily add up to 1, and the remaining probability is interpreted as uncertainty. In the following, we present an overview of subjective logic.

Overview of subjective logic

If a person needs to express her agreement on a proposition using binary logic, she might say *agree* or *disagree*. In probabilistic logic, she provides a value between 0 and 1 to express the degree of her agreement on this proposition. In previous logic, a given person cannot express her ignorance or her degree of uncertainty. Subjective logic proposes a solution

for expressing uncertainty. It uses *opinions* as input and output variables. Opinions explicitly express uncertainty about probability values, and can express degrees of ignorance about a subject.

In the terminology of subjective logic, an opinion held by an individual P about a proposition x is the ordered quadruple $O_x = (b_x, d_x, u_x, a_x)$ where:

- b_x (belief) is the belief that x is true.
- d_x (disbelief) is the belief that the x is false.
- u_x (uncertainty) is the amount of uncommitted belief.
- a_x is called the base rate, it is the a priori probability in the absence of evidence.

With $b_x, d_x, u_x, a_x \in [0..1]$ and $b_x + d_x + u_x = 1$.

The opinion's probability expectation $E(O_x) = b_x + a_x u_x$ is interpreted as a probability measure indicating how x is expected to behave in the future. a_x is used for computing $E(O_x)$. More precisely, a_x determines how uncertainty shall contribute to the probability expectation value $E(O_x)$ [Jøso1].

Subjective Logic is directly compatible with traditional mathematical frameworks as we show in the following:

- If $b = 1$, that is equivalent to binary logic "true".
- If $d = 1$, that is equivalent to binary logic "false".
- If $b + d = 1$, that is equivalent to classical probability constraint.

In subjective logic, if $b + d < 1$, it means degrees of uncertainty represented by the value of $u = 1 - b - d$, and if $b + d = 0$, it means total uncertainty.

An opinion O_x can be defined as a point in the tri-dimensional bounded space shown in Figure 3.3-(a). The belief axis b_x , the disbelief axis d_x and the uncertainty axis u_x run from the middle point of one edge to the opposite corner. In Figure 3.3-(b), the horizontal bottom line between the belief and disbelief corners represents opinion's probability expectation $E(O_x)$. The base rate is represented as a point on the probability axis. The line joining the top corner of the triangle and the base rate point is called the director. The value $E(O_x)$ is formed by projecting the opinion point onto the probability axis in parallel to the base rate director line. For instance, in Figures 3.3-(a),3.3(b), the point O_x represents the opinion $O_x = (0.4, 0.1, 0.5, 0.6)$ and $E(O_x) = 0.7$.

Subjective logic consists of a set of logical operations like the conjunction, the disjunction, the consensus and the discounting operators which are defined to combine opinions³. As we see, in subjective logic users can express their uncertainty about a proposition. This makes it appropriate to evaluate trust.

³These operators are presented in detail in Chapter 6 in Section 6.2

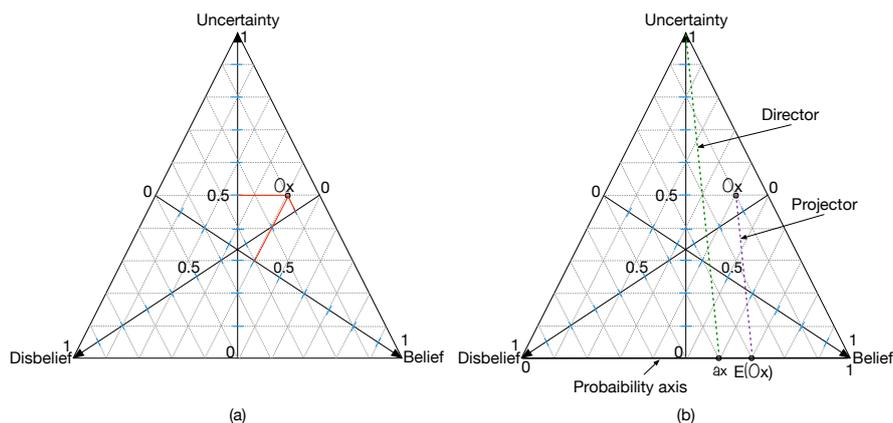


Figure 3.3 – *Opinion space*.

3.2.3 Local vs global trust

Trust can be classified into global or local value. The global trust value represents a single trust value in a subject regardless of who asks for this value. Essentially, for each subject, one trust value is computed. For instance, reputation systems compute a global trust value. Ebay is an example of these systems because it calculates a single trust value for each of its clients.

The local trust value represents the point of view of the person who asks for this value like most of the trust works in social networks [Golo5]. Two persons who ask for a trust value of one subject, they might obtain two different values.

In [MA05], authors make a study on data from the real and large user community, Epinions.com. It uses computational experiments to investigate the differences between global and local trust. The experiments they conducted show that a local trust achieves higher accuracy than a global one in predicting the trust a specific user should place into another user.

There are many proposed approaches for evaluating trust in the literature [ABAR12, AABR13, Golo5, JBo8, Mar94] and several interesting surveys [AG07, Golo6, JBo7, ZDB11]. The approaches we present in the next section are graph-based.

3.3 GRAPH-BASED APPROACHES FOR EVALUATING TRUST

Trust approaches based on graphs [Golo5, Golo6, HWS09, JBo8, JHP06, LW11a, RAD03] are especially used in social networks where the main idea of trust derivation is to propagate it between two nodes in a graph that represents the social network.

3.3.1 Trust propagation

A social network is a social structure made up of a set of persons (individuals or organizations) and a set of relations between these persons. It can be represented as a graph where the nodes are the persons and the edges are the relations between them. Trust between two persons in a social network can be evaluated based on this graph where the source node

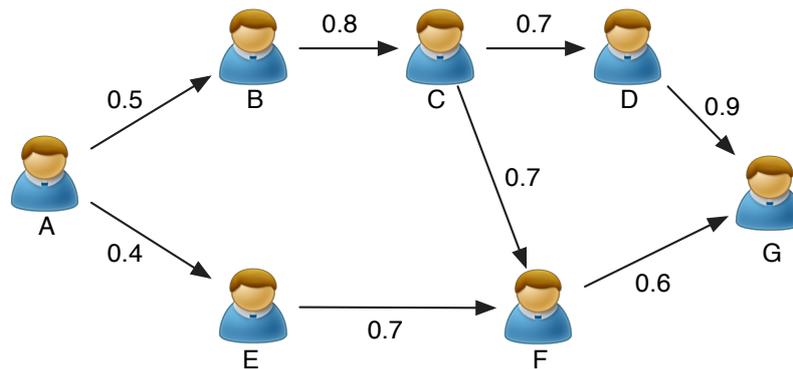


Figure 3.4 – Trust relationships in social network.

is the trustor, the target node is the trustee and the other nodes are the intermediate nodes between the trustor and the trustee. Values are associated to the edges to represent the trust value attributed by the edge source node towards the edge target node. Figure 3.4 shows an example of trust relationships in a social network where, for instance, B trusts C with the value 0.8.

Trust propagation focuses on finding a trust value in a defined person or resource through the multiple paths that relate the source with the target. For instance, in Figure 3.4, how much A trusts G knowing that there are different paths that relate A with G . The paths are: $path_1 = \{A, B, C, D, G\}$, $path_2 = \{A, B, C, F, G\}$ and $path_3 = \{A, E, F, G\}$. To propagate trust through a graph, two steps are considered [AFGLo8]:

- Transitive trust combination through a path: the main idea of this step is to compute a trust value in a *target node* through a path by combining the trust values among the intermediate nodes that belong to this path. In simple words, this step focuses on the problem of trust transitivity through a path, if A trusts B and B trusts C , does A trust C ? and how much? Several operators are employed for trust transitivity ranging from basic operators like the minimum or new proposed operators like the *discounting* operator in subjective logic.
- Parallel trust combination through a graph: the main idea of this step is to combine the several trust values through the multiple paths that relate the source with the target to obtain a single trust value in a *target node* through the whole graph. For instance, if σ_1 , σ_2 and σ_3 are paths that relate A with G and A trusts G with different values through these paths, how much A trusts G through the whole graph composed of σ_1 , σ_2 and σ_3 . Several operators are employed for trust parallel combination through a graph ranging from basic operators like the average or new proposed operators like the *consensus* operator in subjective logic.

In the following, we present some of these studies.

Richardson *et al.* [RADo3] discuss an abstract framework for trust propagation. In this approach, the previous two phases are followed to obtain trust values through the paths that relate the trustor with the trustee using the multiplication operator, then a single value is obtained through

a graph that contains several paths. The proposed operators to obtain this value are the minimum, the maximum or the average operators. Authors argue that the ideal operator is a user-dependent.

Golbeck [Gol05] proposes two types of methods for trust propagation. The first one, [GH06] considers binary trust values. Firstly, the source node classifies all intermediate nodes either as trustworthy or non-trustworthy. The trustworthy nodes, from the point of view of a source node, are the nodes that agree with the source with a certain probability on the classification of the nodes, while non-trustworthy nodes mostly disagree with the opinion of the source node. Since the associated value of trust to each node is either trust or distrust, the trust propagation occurs through the trustworthy paths only (*i.e.*, the paths that contain trustworthy nodes only). This can be considered as a kind of graph simplification that is based on eliminating the paths that can be considered as unreliable. To compute a trust value towards the target node, the source node polls each of its neighbors to obtain their trust values in the target. If a neighbor N has a direct trust relationship with the target, its trust value is returned. If N does not have a direct trust value in the target, it queries all of its neighbors for their trust values. Each neighbor repeats this process till reaching the target. Once this search is complete, the source averages the received values and rounds the final value. This rounded value is the inferred trust value in the target.

The second method proposed by Golbeck, named *TidalTrust*, considers discrete trust values between 1 and 10. The source node associates the nodes that have a direct relationship with the target with a value of credibility. The previous algorithm is applied except the last step. Instead of computing the rounded average value, the source node computes a weighted average value. The weight is defined by the value of credibility.

By observing the different approaches that are proposed to study the trust propagation through a graph, Agudo *et al.* [AFGL08] propose a general model for trust propagation using discrete metrics. They split the process of computing trust in two steps. Firstly, they introduce an abstract sequential operator to compute a trust value through a path. Secondly, they introduce an abstract parallel operator to obtain a value of trust through the whole graph. By making an overview on the proposed operators in the literature, they propose the following combination of (sequential, parallel) operators: (min, min), (min, max), (product, min), (product, max), (product, mean).

To summarize, two levels of computing should be considered when propagating trust through a graph. The first one is computing a trust value in a target node through a path. The second one is computing a trust value in a target node through a graph to come up with a final trust value. The different proposed methods agree on these phases but they differ in some points, like the graph simplification before applying these two levels of computing, the proposed operators and the navigation algorithm through a graph. Figure 3.5 shows the general method for trust propagation through a graph.

Next section studies the continuous metrics and illustrates a problem for evaluating trust based on a graph using some discrete or continuous metrics.

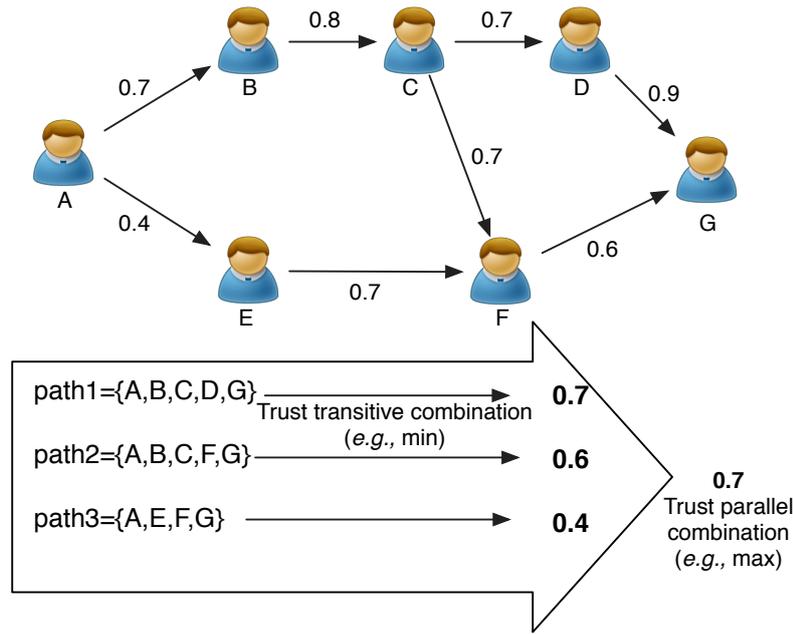


Figure 3.5 – A general method for trust propagation through a graph.

3.3.2 Dependent paths: a potential problem

In [JHP06, JBo8], Jøsang *et al.* raised a problem of previous approaches. They argue that some computational models do not give exact results when there are dependent paths (*i.e.*, paths that have common edges) in the graph.

To explicit this problem, we give a simple example shown in Figure 3.6. We need to evaluate T_E^A , A's trust value in E. The paths between A and E are: $path_1 = \{A, B, C, E\}$ and $path_2 = \{A, B, D, E\}$. There is a common edge between these two paths which is $A \rightarrow B$.

Let \otimes be the operator of trust transitive combination through a path and \oplus be the operator of trust parallel combination through a graph. To evaluate T_E^A , the A's trust value in E:

$$T_E^A = T_B^A \otimes ((T_C^B \otimes T_E^C) \oplus (T_D^B \otimes T_E^D)) \quad (3.1)$$

However, if we apply the general method presented in Section 3.3, T_E^A is computed as follows:

$$T_E^A = (T_B^A \otimes T_C^B \otimes T_E^C) \oplus (T_B^A \otimes T_D^B \otimes T_E^D) \quad (3.2)$$

Relations 3.1, 3.2 consist of the same two paths $path_1$ and $path_2$, but their combined structures are different. T_B^A appears twice in Relation 3.2. In some computational models the previous two equations produce different results depending on which expression is being used. Whereas in some other model, the results would be the same. For instance, when implementing \otimes as binary logic “and”, and the parallel \oplus as binary logic “or”, the results would be equal. However, when implementing \otimes and \oplus as probabilistic multiplication and comultiplication respectively, the results would be different. It is also different in the case of applying the discounting and consensus operators in subjective logic for transitivity

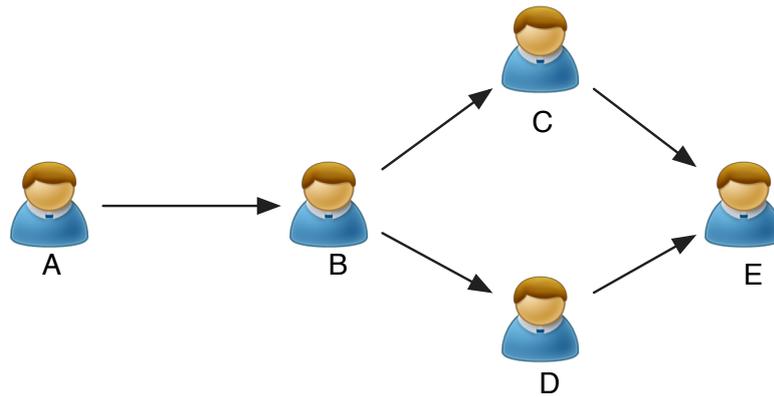


Figure 3.6 – The problem of the graphs that has dependent paths
 $T_B^A \otimes ((T_C^B \otimes T_E^C) \oplus (T_D^B \otimes T_E^D)) \neq (T_B^A \otimes T_C^B \otimes T_E^C) \oplus (T_B^A \otimes T_D^B \otimes T_E^D)$

and parallel combination respectively. If \otimes is the average function and \oplus is the minimum or maximum function, the results are also different.

Table 3.1 shows examples of applying different operators on a simple graph and the different obtained results of Relations 3.1 and 3.2 using some discrete or continuous metrics.

Jøsang *et al.* [JHP06] propose an interesting approach for trust propagation based on subjective logic. Similar to the previous approaches, authors propose a method to combine opinions through a path using an operator called *discounting* in subjective logic that helps to deduce the transitivity of opinions, and to compose opinions through the different paths using an operator called *consensus* in subjective logic that allows to combine several parallel opinions.

To deal with the problem of the multiple calculations of the values associated to common edges, Jøsang *et al.* propose a method based on graph simplification and trust derivation with subjective logic. This approach is called Trust Network Analysis with Subjective Logic (TNA-SL). They simplify a complex trust graph into a graph having independent paths by removing the most uncertain paths *i.e.*, the paths that have a high value of uncertainty (*cf.* Section 3.2.2.2). The left side of Figure 3.7 shows a graph simplification in TNA-SL.

The main disadvantage of TNA-SL is that a complex graph must be simplified before it can be analyzed, which can lead to loss of information. To avoid this loss of information, authors describe a new method for trust network analysis in [JB08]. They split each common edge into as many edges as the number of paths sharing this edge to obtain a graph with independent paths. The opinion assigned to the edge in common is also split in a way that if the split opinions are combined again, they produce the original opinion associated to the original edge. The right side of Figure 3.7 shows the graph transformation to a graph that has independent paths by splitting the edge and the opinion associated to this edge.

To summarize, the trust propagation works mainly follow the same approach for evaluating trust in a target node in a graph by following two steps: (1) trust propagation through a path and (2) trust propagation through a graph employing different metrics and operators (*cf.* Section 3.2.2). Before doing these two steps, some works have proposed a

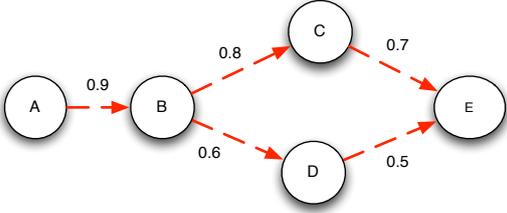
Graph	
Discrete metrics	\otimes average \oplus maximum
	Equation 3.1 $T_E^A = 0.825$
	Equation 3.2 $T_E^A = 0.8$
Continuous metrics	\otimes multiplication \oplus comultiplication
	Equation 3.1 $T_E^A = 0.623$
	Equation 3.2 $T_E^A = 0.64$

Table 3.1 – The different obtained results of Relations 3.1, 3.2 by applying an example of discrete metrics and continuous metrics on a simple graph

graph simplification where they eliminate some paths that are considered as not important. The details of the proposed metrics and operators differ from a work to another. Jøsang is one of the first who discusses the problem of dependent paths explicitly. His proposed solutions work only for the trust propagation problem using subjective logic. The other approaches, either ignore the problem [RAD03, AFGL08], either propose simple solutions like choosing one path in a graph [LWOL11], or removing the paths that are considered unreliable [GH06, JHP06].

3.4 CONCLUSION

Trust represents now a huge field that contains several and varied aspects. One of the objectives of this thesis is evaluating trust in a system for an activity. To fulfill this objective, we noticed that some main concepts about trust should be studied and adopted like trust definition (*cf.* Section 3.2.1), metrics (*cf.* Section 3.2.2) and the type of trust that should be evaluated (local or global) (*cf.* Section 3.2.3).

In this thesis, we adopt the definition proposed by Jøsang about trust “trust is the subjective probability, by which an individual expects that another individual performs a given action on which its welfare depends”. This definition, which is adopted from social sciences, is simple and realistic, it has a

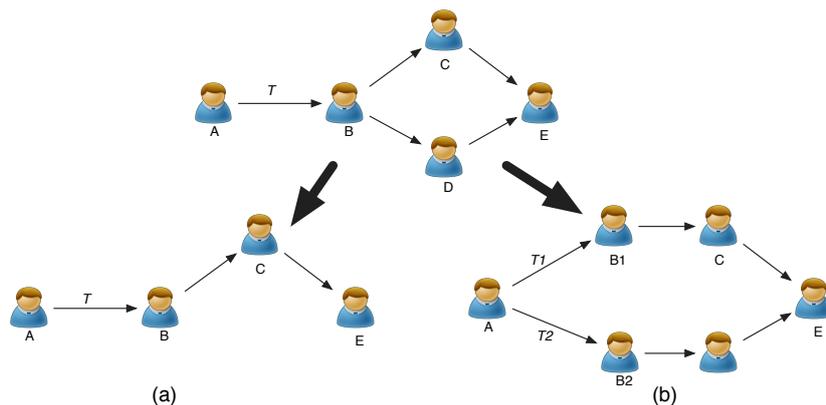


Figure 3.7 – Proposed solutions by Jøsang et al. to transform a graph having dependent paths to a graph having independent paths.

(a) Eliminating the uncertain paths (TNA-SL).

(b) Splitting the common edges and the opinions associated to them, $T = T_1 \oplus T_2$.

direct relevance to our context and it can be interpreted as a mathematical formalism since trust is defined as a value of probability.

Since trust is defined as a probability in this thesis, the continuous metrics are adopted. We aim to take benefits of the rich body of probabilistic methods to be applied to the context to our work, more precisely probability theory and subjective logic are employed to evaluate trust in this thesis.

Even if most of the current research studies evaluate a global trust value, we share the vision of some researchers [MA05, Gol05] in believing that such a value is misleading and trust evaluations must be made from the perspective of the individual. Thus, in this work we choose to evaluate a local trust value.

We focused on the graph-based trust approaches in social network which converge with our work in relying on a graph to evaluate trust. We both collect trust values through a graph then combining these values. However, the main idea of their works is to compute a trust value in *a target node* in a graph depending on the trust values among the intermediate nodes. In our work, we depend on the trust values associated to the intermediate nodes by a particular user to compute a trust value towards *the graph as a combination of several nodes*.

Furthermore, the interpretation of the graph is different. For us, a graph represents a system for a digital activity achieved by a particular user and not a social network. This assumption plays an important role in the operations we apply and in the results we interpret. Due to the difference in the graph interpretation, the semantics of trust value associated to the graph entities and the different granularities of trust we estimate, we cannot adopt directly their methods or operators. We need to use other methods that are more adaptable to the context of our work.

Moreover, in this chapter, we explicitly introduced the problem of the dependent paths in a graph for evaluating trust. This problem will be treated in our work.

Next chapter presents SOCIOPATH, our proposed metamodel that allows to model a system for an activity.

SOCIOPATH: MODELING A SYSTEM

4

“It is impossible to work in information technology without also engaging in social engineering.”

-Jaron Lanier.

CONTENTS

5.1	INTRODUCTION	53
5.2	A SOCIOPATH MODEL AS A DIRECTED ACYCLIC GRAPH (DAG)	54
5.3	A PROBABILISTIC APPROACH TO INFER SYSTEM TRUST VALUE	55
5.3.1	Trust in a node for an activity	56
5.3.2	Trust in a path for an activity	56
5.3.3	Trust in a system for an activity	57
5.4	EXPERIMENTAL EVALUATIONS	59
5.4.1	Influence of the system architecture on the trust value	59
5.4.2	Influence of the path length and the number of paths on the trust value	60
5.4.3	Social evaluation: a real case	61
5.5	CONCLUSION	63

IN the EA, the modeling covers all the complexity of an enterprise architecture including the financial aspects, processes, projects, *etc.* In our work, we need a simple representation of a system for an activity that reveals some aspects about the used architecture like the relations of dependences. These relations will be used for evaluating trust. This imposes different notions in the modeling, different degrees of complexity and different approaches to follow, which are inspired from the concepts presented in the EA works. In this chapter, we present the first contribution of this PhD thesis: the SOCIOPATH metamodel. SOCIOPATH allows to draw a representation (or model) of a system that identifies its hardware, software and persons as components, and the ways they are related (*cf.*

Section 4.2). Enriched with deduction rules (*cf.* Section 4.3), SOCIOPATH analyzes the relations between the components and deduces some implicit relations. In SOCIOPATH, we propose some definitions that reveal main aspects about the used architecture for a user (*cf.* Section 4.4). An illustrating example is presented in Section 4.5.

4.1 INTRODUCTION

Nowadays, the most widespread architectures belong to the domain of distributed systems. Most of participants' activities on these systems concern their data (sharing and editing documents, publishing photos, purchasing online, *etc.*). Using these systems implies some implicit and explicit relationships which may be partly unknown. Indeed, users achieve several activities without being aware of the used architecture. In our approach, we believe that users need to have a general visual representation of the used system including the social and digital entities. Based on this representation, a lot of implicit relations can be deduced like the relations of the social dependence. By proposing SOCIOPATH [ALB⁺12], we aim to answer the user's questions about its used system:

1. Who are the persons that have a possibility to access a user's data? and what are the potential coalitions among persons that could allow undesired access to this data?
2. Who are the person(s)/resource(s) a user depends on to perform an activity?
3. Who are the persons who can prevent a user from performing an activity?
4. Who are the persons whom a user is able to avoid to perform an activity?
5. How much a user trusts a system for a specific activity?

Some of these questions raise several issues as someone may be able to grab information about who the user is, what the user does, and so forth. That directly leads to privacy [Wes70], trust [MGMo6] and security issues [LABW92].

The analysis of such systems is usually limited to technical aspects as latency, QoS, functional performance, failure management [ACH96], *etc.* The aforementioned questions give some orthogonal but complementary criteria to the classical approach. Currently, people underestimate dependences generated by the systems they use and the resulting potential risks.

In this chapter, inspired by the enterprise architecture, we propose the SOCIOPATH metamodel. This approach is based on notions coming from many fields, ranging from computer science to sociology. SOCIOPATH is a generic metamodel that is divided in two worlds; the social world and the digital world. SOCIOPATH allows to draw a representation (or model) of a system that identifies its hardware, software and persons as components, and the ways they are related. Enriched with deduction rules, SOCIOPATH analyzes the relations between the components and deduces some implicit relations according to a specific activity concerning some data.

4.2 SOCIOPATH METAMODEL

The SOCIOPATH metamodel allows to describe the architecture of a system in terms of the components that enable people to access digital resources.

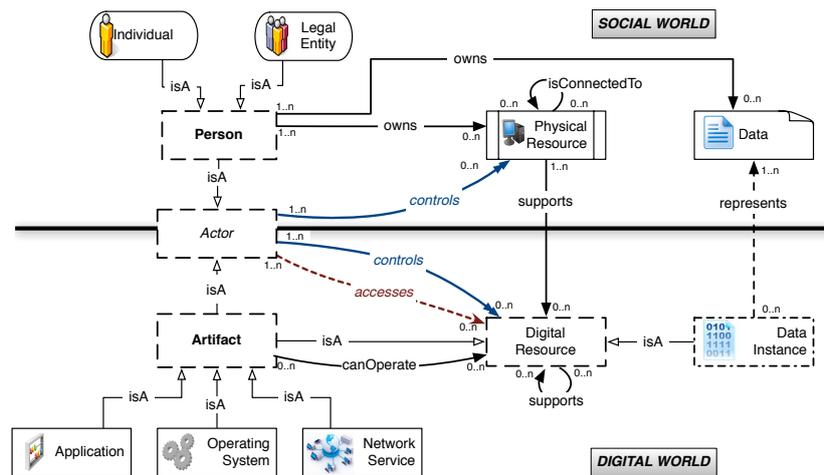


Figure 4.1 – Graphical view of SOCIOPATH as a UML class diagram.

It distinguishes two worlds; the *social world* and the *digital world*, the social world where persons or organizations own any kind of physical resources and data, and the digital world where instances of data (including source codes) are stored and processes are running. Figure 4.1 shows the graphical representation of SOCIOPATH, that we analyze in this section.

4.2.1 The social world

The social world includes persons (users, enterprises, companies, etc.), physical resources, data and relations among them.

- *Data* represent an abstract notion that exists in real life, and does not necessarily imply a physical instance (e.g., address, age, software design, etc.).
- *Physical Resource* represents any hardware device (e.g., PC, USB device, etc.).
- *Person* represents a generic notion that defines an Individual like Alice or a Legal Entity like Microsoft.

4.2.2 The digital world

The digital world has entities that are defined as follows:

- *Data Instance* represents a digital representation of *Data* that exist in the social world. For instance, a person has an address (*Data*) in the social world. Whenever she writes it in an object, she creates a semantically equivalent digital instance of her address in the digital world (*Data Instance*). In that way, the source code of a software is a representation of the software design in the digital world.
- *Artifact* represents an abstract notion that describes a “running software”. This can be an *Application*, an *Operating System* or a *Network Service*. It may be a single process or a group of processes that should be distributed on different locations, yet defining a single logically coherent entity.

- *Digital Resource* represents an *Artifact* or a *Data Instance*.
- *Actor* represents a *Person* in the social world or an *Artifact* in the digital world. This is the core concept of SOCIOPATH. Indeed, only *Actors* can access or control *Digital Resources* as presented below.

4.2.3 The relations in SocioPath

Several relations are drawn in SOCIOPATH. In this section, we briefly describe them.

- *owns* represents a relation of ownership between a *Person* and a *Physical Resource* ($owns(P, PR)$), or between a *Person* and some *Data* ($owns(P, D)$). This relation only exists in the social world.
- *isConnectedTo* represents a relation of connection between two *Physical Resources* ($isConnectedTo(PR_1, PR_2)$). It means that two entities are physically connected, through a network for instance. This symmetric relation exists only in the social world.
- *canOperate* represents an *Artifact* that is able to process, communicate or interact with a target *Digital Resource* ($canOperate(F, DR)$). This ability may be explicitly given. For instance, “Microsoft Word” *canOperate* a `.doc` document or deduced from some property of the system, for instance, an operating system only *canOperate* files that are stored in a mounted partition.
- *accesses* represents an *Actor* that can access a *Digital Resource* ($accesses(A, DR)$). For instance, the operating system accesses the applications installed via this operating system; a person who owns a PC that supports an operating system accesses this operating system. The access relations we consider are: read, write, execute.
- *controls* represents an *Actor* that can control a *Digital Resource* ($controls(A, DR)$). There should exist different kinds of control relations. For instance, a legal entity, who provides a resource, controls the functionalities of this resource. The persons who use this resource may have some kind of control on it as well. Each of these actors controls the resource in a different way.
- *supports* represents a relation between two *Digital Resources* ($supports(DR_1, DR_2)$), or a *Physical Resource* and a *Digital Resource* ($supports(PR, DR)$). It means that the target entity could never exist without the source entity. We may say that the latter allows the former to exist. For instance, an operating system is supported by a given hardware; an application is supported by an operating system; the code of an application supports this application.
- *represents* represents a relation between *Data* in the social world and their *Instances* in the digital world ($represents(D, DI)$). For instance, the source code of the operating system Windows is a representation in the digital world of the data known as “Microsoft Windows” in the social world.

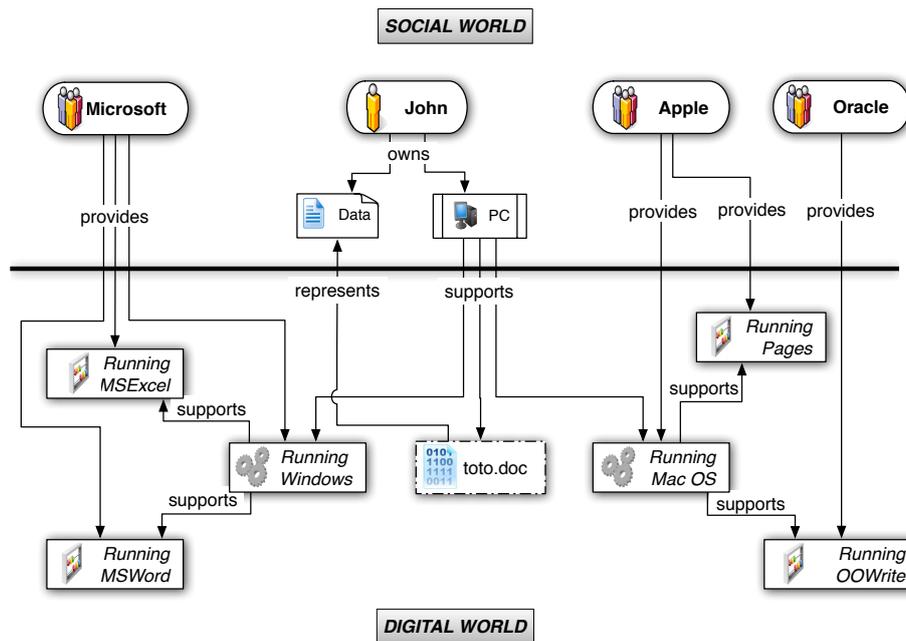


Figure 4.2 – Use case example: a document accessed by 2 different operating systems.

Notice that most of the above relations are not symmetrical. The cardinality of the relations is given in Figure 4.1. Considering the relation *owns* as an example, a *Person* can own some *Physical Resources*, i.e., $[0..n]$, and every *Physical Resource* is owned by at least one *Person*, i.e., $[1..n]$.

Persons own some data in the social world. *Data* have a concrete existence in the digital world if they are represented by some *Data Instance* and supported by some *Physical Resource*. As an *Actor* in the digital world, a *Person* can access and control *Data Instances* representing her (and others) *Data*. This may be done through different resources, thus implying some dependences on other persons.

Moreover, we consider that a person *provides* an artifact (cf. Figure 4.3) if this person owns data represented by a data instance which supports the artifact.

Applying SOCIOPATH makes possible non-trivial deductions about relations among entities. For instance, an actor may be able to access digital resources supported by different physical resources connected to each other (e.g., a user can access processes running in different hosts).

4.2.4 Example of a SocioPath model: single PC

Figure 4.2 shows a basic SOCIOPATH model¹ of a use-case on a unique PC. In the social world, a user *John* owns some *Data* and a *PC*. There are also legal entities as: *Microsoft*, provider of *Windows*, *Microsoft Word* (*MSWord*) and *Microsoft Excel* (*MSEExcel*); *Apple*, provider of *MacOS* and *Pages*; and *Oracle*, provider of *Open Office Writer* (*OOWrite*).

In the digital world, two operating systems exist on *John's PC*: *Windows* and *MacOS*. On *Windows*, two applications are available: *MSWord* and

¹In general, we consider that a model conforms to a metamodel.

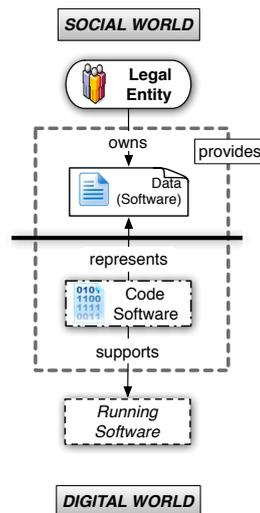


Figure 4.3 – The relation provide.

Concept	Notation	Set	Remark
Actor	A	\mathbb{A}	$A \in \mathbb{A}$
Artifact	F	\mathbb{F}	$F \in \mathbb{F}$
Digital resource	DR	\mathbb{DR}	$DR \in \mathbb{DR}$
Physical resource	PR	\mathbb{PR}	$PR \in \mathbb{PR}$
Data	D	\mathbb{D}	$D \in \mathbb{D}$
Data instance	DI	\mathbb{DI}	$DI \in \mathbb{DI}$
Operating system	OS	\mathbb{O}	$OS \in \mathbb{O}$
Path	σ	\mathbb{Y}	$\sigma \in \mathbb{Y}$
Architecture or system	α	\mathbb{A}	$\alpha \in \mathbb{A}$
Activity	ω	\mathbb{W}	$\omega \in \mathbb{W}$
Activity path	σ^ω	\mathbb{Y}^ω	$\sigma^\omega \in \mathbb{Y}^\omega$
Activity minimal path	$\widehat{\sigma^\omega}$	$\widehat{\mathbb{Y}^\omega}$	$\widehat{\sigma^\omega} \in \widehat{\mathbb{Y}^\omega}$
Set of activity restrictions	\mathcal{S}	\mathbb{S}	$\mathcal{S} \in \mathbb{S}$
Person or user	P	\mathbb{P}	$P \in \mathbb{P}$

Table 4.1 – Glossary of notations (1).

MSExcel. On MacOS are installed OOWrite and Pages. John's Data are represented in the digital world by the document `toto.doc`.

We use this example to illustrate some deductions in Section 4.3. We deliberately propose a trivial example, in order to show how SOCIOPATH can be applied and how some deductions and definitions are drawn. Table 4.1 summarizes the notations we use in the following.

4.3 DEDUCED ACCESS AND CONTROL RELATIONS

The semantics of the components and the relations of a SOCIOPATH model allows to deduce more *controls* and *accesses* relations. We use ProLog, a First Order Logic (FOL) language to describe the rules allowing such deductions. Thus, in the next, the SOCIOPATH components correspond to constants and the relations are described by binary predicates. For in-

stance, $supports(OS, F)$ represents the *supports* relation between the operating system OS and the artifact F .

The proposed deduction rules of SocioPath are not exhaustive and by no means we pretend they capture the whole complexity of systems. They capture several aspects of a simplified vision of the systems that serves the purpose of building an understandable and expressive model.

- **Rule 1:** If an artifact can operate a digital resource and either the artifact and the digital resource are supported by the same physical resource or they are supported by connected physical resources, then the artifact accesses the digital resource.

$$\forall F \in \mathbb{F}, \forall DR \in \mathbb{DR}, \forall PR1, PR2 \in \mathbb{PR} :$$

$$\bigwedge \left\{ \begin{array}{l} canOperate(F, DR) \\ supports(PR1, F) \end{array} \right. \Rightarrow accesses(F, DR)$$

$$\vee \left\{ \begin{array}{l} supports(PR1, DR) \\ \bigwedge \left\{ \begin{array}{l} supports(PR2, DR) \\ isConnectedTo(PR1, PR2) \end{array} \right. \end{array} \right.$$

E.g., Windows accesses MSWord:

$$\bigwedge \left\{ \begin{array}{l} canOperate(Windows, MSWord) \\ supports(PC, Windows) \\ supports(PC, MSWord) \end{array} \right. \Rightarrow accesses(Windows, MSWord)$$

- **Rule 2:** If a person owns a physical resource that supports an operating system, then the person accesses and controls this operating system.

$$\forall P \in \mathbb{P}, \forall PR \in \mathbb{PR}, \forall OS \in \mathbb{O} :$$

$$\bigwedge \left\{ \begin{array}{l} owns(P, PR) \\ supports(PR, OS) \end{array} \right. \Rightarrow \bigwedge \left\{ \begin{array}{l} accesses(P, OS) \\ controls(P, OS) \end{array} \right.$$

E.g., John accesses and controls Windows:

$$\bigwedge \left\{ \begin{array}{l} owns(John, PC) \\ supports(PC, Windows) \end{array} \right. \Rightarrow \bigwedge \left\{ \begin{array}{l} accesses(John, Windows) \\ controls(John, Windows) \end{array} \right.$$

- **Rule 3:** If an operating system supports and can operate an artifact, then it controls this artifact.

$$\forall F \in \mathbb{F}, \forall OS \in \mathbb{O} : \bigwedge \left\{ \begin{array}{l} supports(OS, F) \\ canOperate(OS, F) \end{array} \right. \Rightarrow controls(OS, F)$$

E.g., `Windows` controls `MSWord`:

$$\bigwedge \left\{ \begin{array}{l} \text{supports}(\text{Windows}, \text{MSWord}) \\ \text{canOperate}(\text{Windows}, \text{MSWord}) \end{array} \right\} \Rightarrow \text{controls}(\text{Windows}, \text{MSWord})$$

- **Rule 4:** If a person owns data represented in the digital world by a data instance which supports an artifact, then this person controls this artifact.

$\exists P \in \mathbb{P}, \exists D \in \mathbb{D}, \exists DI \in \mathbb{DI}, \exists F \in \mathbb{F} :$

$$\bigwedge \left\{ \begin{array}{l} \text{owns}(P, D) \\ \text{represents}(DI, D) \\ \text{supports}(DI, F) \end{array} \right\} \Rightarrow \text{controls}(P, F)$$

E.g., Microsoft owns the data `Windows` which is represented in the digital world on John's PC by the data instance `CodeWindows` which supports the application `Windows`, so Microsoft controls `Windows`:

$$\bigwedge \left\{ \begin{array}{l} \text{owns}(\text{Microsoft}, \text{Data-Windows}) \\ \text{represents}(\text{CodeWindows}, \text{Data-Windows}) \\ \text{supports}(\text{CodeWindows}, \text{Windows}) \end{array} \right\} \Rightarrow \text{controls}(\text{Microsoft}, \text{Windows})$$

- **Rule 5:** The relation *accesses* is transitive. We denote the relation *accesses*, which is deduced from the transitivity, with *accesses**.

$\forall A \in \mathbb{A}, \forall F \in \mathbb{F}, \forall DR \in \mathbb{DR} :$

$$\bigwedge \left\{ \begin{array}{l} \text{accesses}(A, F) \\ \text{accesses}(F, DR) \end{array} \right\} \Rightarrow \text{accesses}^*(A, DR)$$

E.g., `MSWord` accesses `Windows` and `Windows` accesses `toto.doc`, so `MSWord` *accesses** `toto.doc`:

$$\bigwedge \left\{ \begin{array}{l} \text{accesses}(\text{MSWord}, \text{Windows}) \\ \text{accesses}(\text{Windows}, \text{toto.doc}) \end{array} \right\} \Rightarrow \text{accesses}^*(\text{MSWord}, \text{toto.doc})$$

- **Rule 6:** The relation *controls* is transitive. We denote the relation *controls*, which is deduced from the transitivity, with *controls**.

$\forall A \in \mathbb{A}, \forall F_1, F_2 \in \mathbb{F} :$

$$\bigwedge \left\{ \begin{array}{l} \text{controls}(A, F_1) \\ \text{controls}(F_1, F_2) \end{array} \right\} \Rightarrow \text{controls}^*(A, F_2)$$

E.g., John controls `Windows` and `Windows` controls `toto.doc` so John *controls** `toto.doc`:

$$\bigwedge \left\{ \begin{array}{l} \text{controls}(\text{John}, \text{Windows}) \\ \text{controls}(\text{Windows}, \text{toto.doc}) \end{array} \right\} \Rightarrow \text{controls}^*(\text{John}, \text{toto.doc})$$

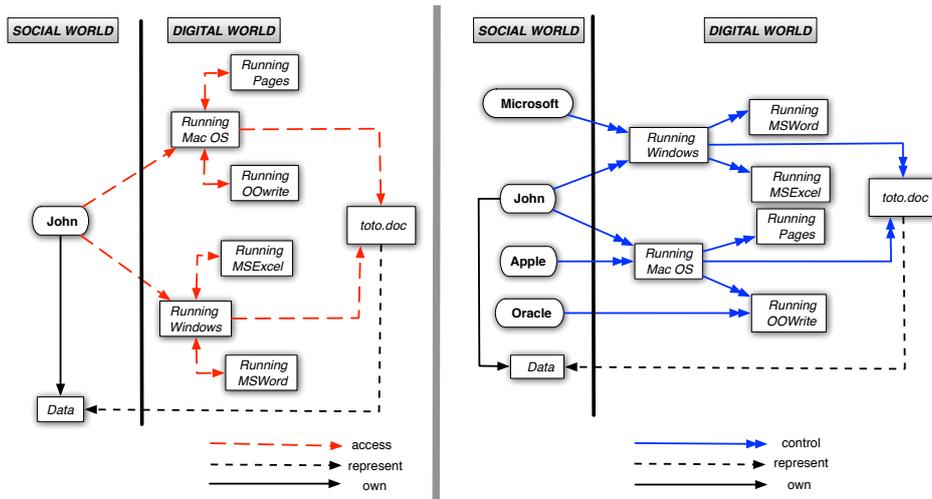


Figure 4.4 – The relations of access/control in the example: a document accessed by 2 different operating systems.

- **Rule 7:** If two physical resources are connected to each other, and the first one supports an operating system and the second one supports another operating system, these two operating systems access to each other. $\exists PR1, PR2 \in \mathbb{P}\mathbb{R}, \exists OS1, OS2 \in \mathbb{O}$:

$$\bigwedge \left\{ \begin{array}{l} isConnectedTo(PR1, PR2) \\ supports(PR1, OS1) \\ supports(PR2, OS2) \end{array} \right. \Rightarrow \bigwedge \left\{ \begin{array}{l} accesses(OS1, OS2) \\ accesses(OS2, OS1) \end{array} \right.$$

Starting from the example of Section 4.2.4, we apply the SocioPath rules, and obtain the *accesses* and *controls* relations shown in Figure 4.4. Thus, for example, from Rule 2, we deduce that John accesses and controls the operating systems MacOS and Windows, and from Rule 4, we deduce that Microsoft controls the operating system Windows and Apple controls the operating system MacOS.

The following are two examples of the programmed rules in ProLog.

```
accesses(Person, OS) :-
    operatingSystem(OS),
    person(Person),
    owns(Person, Computer),
    supports(Computer, OS).
```

Rule 2 in ProLog.

```
controls(Person, DResource) :-
    person(Person),
    dResource(DResource),
    data(Data),
    dataInstance(DInstance),
    owns(Person, Data),
    represents(DInstance, Data),
    supports(DInstance, DResource).
```

Rule 4 in ProLog.

4.4 SOCIOPATH DEFINITIONS

We enrich SocioPath with several formal definitions presented in the following sections. With these definitions, we give answers to the questions 1 to 4 presented in the introduction of this chapter.

4.4.1 Activities and paths

A user follows a path to perform an activity in a system. Here, we consider activities involving data (e.g., copying a file, sharing a document, etc.). This means that some restrictions must be given to the ways the person might do their activity. We can do this by imposing the presence of particular elements in the path to do the activity. For instance, if a person wants to read a `.doc` document, she must use an artifact that can “understand” this type of document (e.g., `MSWord` or `OOWrite`). Other example, if a person uses a SVN application, the artifacts “SVN client” and “SVN server” should be used and they should appear in the correct order within the path (usually, the SVN client should precede the SVN server).

Definition 4.1 (Activity).

We define an activity ω as a triple (P, D, S) , where P is a person, D is data and S is a set of ordered sets of \mathbb{F} in a model. So an activity ω is a subset of $\mathbb{P} \times \mathbb{D} \times \mathbb{S}$. The sets in the S component of an activity are alternative sets of artifacts that one of them is necessary to perform an activity. Thus, $\omega = (P, D, S) \in \mathbb{P} \times \mathbb{D} \times \mathbb{S}$. For instance, the activity “John edits toto.doc” is defined as $\omega = (\text{John}, \text{Data}, \{\{\text{MSWord}\}, \{\text{Pages}\}, \{\text{OOWrite}\}\})$.

We call *paths* the lists of actors and digital resources that describe the ways an actor may access a digital resource. A person may perform an activity in different ways and using different intermediate digital resources. Each possibility can be described by a path.

Definition 4.2 (Activity path, or ω -path).

A path σ for an activity $\omega = (P, D, S) \in \mathbb{P} \times \mathbb{D} \times \mathbb{S}$ is a list of actors and digital resources such that:

- $\sigma[1] = P$;
- $\sigma[|\sigma|] = D$;
- *represents* $(\sigma[|\sigma| - 1], \sigma[|\sigma|])$;
- $\forall i \in [2 : |\sigma| - 1], (\sigma[i] \in \mathbb{F}) \wedge \text{accesses}(\sigma[i - 1], \sigma[i])$;
- $\exists s \in S, s \subseteq \sigma$;

where $\sigma[i]$, denotes the i^{th} element of σ , and $|\sigma|$ the length of σ .

Notation: Assuming that there is no ambiguity on the model under consideration, the set of ω -paths where $\omega = (P, D, S)$ is noted Y^ω and the set of all the paths in the model is noted Y .

For example, If John wants to achieve the activity $\omega =$ “John edits toto.doc” using the architecture shown in Figure 4.2, John uses Windows to work on the application `MSWord` which uses Windows file system to access the file `toto.doc` so one of the ω -paths for this activity is:

```
{John, Windows, MSWord, Windows, MSExcel, Windows, toto.doc}
```

This path contains some unnecessary artifacts. For instance, `MSExcel` is an unnecessary artifact to edit `toto.doc`. It appears in the ω -path because there exists a relation *accesses* between it and the artifact `Windows`. We want to eliminate all the unnecessary elements from the ω -paths, so we define the activity minimal paths as follows.

Definition 4.3 (Activity minimal path, or ω -minimal path).

Let Y^ω be a set of paths for an activity ω .

A path $\sigma^\omega \in Y^\omega$ is said to be minimal in Y^ω iff there exists no path $\sigma' \in Y^\omega$ such that:

- $\sigma^\omega[1] = \sigma'[1]$ and $|\sigma^\omega| < |\sigma'|$;
- $\forall i \in [2 : |\sigma'|], \exists j \in [2 : |\sigma^\omega|], \sigma'[i] = \sigma^\omega[j]$.
- $\forall i \in [2 : |\sigma| - 1], \text{accesses}(\sigma[i - 1], \sigma[i])$;

Notation: The set of minimal paths enabling an activity $\omega = (P, D, S)$ is noted \widehat{Y}^ω . This set represents also an architecture for an activity α . For sake of simplicity, we name this set the ω -minimal paths.

For instance, for the activity $\omega = \text{“John edits toto.doc”}$, the path `{John, Windows, MSWord, Windows, MSExcel, Windows, toto.doc}` has been eliminated because there is a path $\sigma' = \text{“John, Windows, MSWord, toto.doc”}$ that satisfies the previous conditions. The set of the ω -minimal paths for this activity are:

$$\alpha = \widehat{Y}^\omega = \left\{ \begin{array}{l} \{\text{John, Windows, MSWord, toto.doc}\} \\ \{\text{John, MacOS, OOWrite, toto.doc}\} \\ \{\text{John, MacOS, Pages, toto.doc}\} \end{array} \right\}.$$

4.4.2 Dependence

Modeling systems with `SOCIOPATH` allows to underline and discover chains of *accesses* and *controls* relations. In the following, we want to use these relations and the definitions of Section 4.4.1 to introduce the definitions of digital dependences (Definitions 4.4 and 4.5) and social dependences (Definitions 4.6 to 4.9). Informally, the sets of digital dependences of a person are composed of the artifacts a user passes by to reach a particular element. The sets of social dependences are composed of the persons who control these artifacts and the physical resources that support them. We call the sets of artifacts, which a user depends on, digital dependences because the artifacts belong to the digital world in the `SOCIOPATH` model and we call the sets of persons and physical resources, which a user depends on, social dependences because these two belong to the social world in the `SOCIOPATH` model. In the following, all those concepts are defined formally and the examples refer to Figure 4.2.

4.4.2.1 Digital dependences

We say that a person depends on a set of artifacts for an activity ω if each element of this set belongs to one or more paths in the set of the ω -minimal paths.

Definition 4.4 (Person's dependence on a set of artifacts for an activity).

Let $\omega = (P, D, S)$ be an activity, \mathcal{F} be a set of artifacts and \widehat{Y}^ω be the set of ω -minimal paths.

$$P \text{ depends on } \mathcal{F} \text{ for } \omega \text{ iff } \wedge \left\{ \begin{array}{l} \mathcal{F} \subset \mathbb{F} \\ \forall F \in \mathcal{F}, \exists \sigma \in \widehat{Y}^\omega : F \in \sigma \end{array} \right.$$

For instance, one of the sets on which John depends for the activity "John edits `toto.doc`" is `{MacOS, MSWord}` in Figure 4.2.

A person does not depend on all the sets of artifacts in the same way. Some sets may be avoidable *i.e.*, the activity can be executed without them. Some sets are unavoidable *i.e.*, the activity cannot be performed without them. To distinguish the way a person depends on artifacts, we define the degree of a person's dependence on a set of artifacts for an activity as the ratio of the ω -minimal paths that contain these artifacts to all the ω -minimal paths.

Definition 4.5 (Degree of a person dependence on a set of artifacts for an activity).

Let $\omega = (P, D, S)$ be an activity, \mathcal{F} be a set of artifacts and \widehat{Y}^ω be the set of ω -minimal paths and $|\widehat{Y}^\omega|$ is the number of the ω -minimal paths. The degree of dependence of P on \mathcal{F} , denoted $d_{\mathcal{F}}^\omega$, is:

$$d_{\mathcal{F}}^\omega = \frac{|\{\sigma : \sigma \in \widehat{Y}^\omega \wedge \exists F \in \mathcal{F}, F \in \sigma\}|}{|\widehat{Y}^\omega|}$$

For instance, the degree of dependence of John on the set `{MacOS, MSWord}` for the activity "John edits `toto.doc`" is equal to one, while the degree of dependence of John on the set `{Pages, OOWrite}` is equal to $2/3$.

4.4.2.2 Social dependences

From the digital dependences we can deduce the social dependences as follows. A person depends on a set of persons for an activity if the persons of this set control some of the artifacts the person depends on.

Definition 4.6 (Person's dependence on a set of persons for an activity).

Let $\omega = (P, D, S)$ be an activity, and \mathcal{P} a set of persons.

$$P \text{ depends on } \mathcal{P} \text{ for } \omega \text{ iff } \wedge \left\{ \begin{array}{l} \exists \mathcal{F} \subset \mathbb{F} : P \text{ depends on } \mathcal{F} \text{ for } \omega \\ \forall F \in \mathcal{F}, \exists P' \in \mathcal{P} : \text{controls}(P', F) \end{array} \right.$$

For instance, one of the sets John depends on for the activity "John edits `toto.doc`" is `{Oracle, Apple}` in Figure 4.2.

The degree of a person's dependence on a set of persons for an activity is given by the ratio of the ω -minimal paths that contain artifacts controlled by this set of persons.

Definition 4.7 (Degree of a person's dependence on a set of persons for an activity).

Let $\omega = (P, D, S)$ be an activity, \mathcal{P} be a set of persons and \widehat{Y}^ω be the ω -minimal paths. The degree of dependence of P on \mathcal{P} , noted $d_{\mathcal{P}}^\omega$ is:

$$d_{\mathcal{P}}^\omega = \frac{|\{\sigma : \sigma \in \widehat{Y}^\omega \wedge \exists P' \in \mathcal{P}, \exists F \in \sigma, \text{controls}(P', F)\}|}{|\widehat{Y}^\omega|}$$

For instance, the degree of dependence of John on the set {Oracle, Apple} for the activity “John edits the toto.doc” is equal to 2/3. We recall that Oracle *controls* OOwrite and Apple *controls* MacOS.

We say a person depends on a set of physical resources for an activity if the elements of this set support the artifacts the person depends on.

Definition 4.8 (Person’s dependence on a set of physical resources for an activity).

Let $\omega = (P, D, S)$ be an activity, and \mathcal{PR} be a set of physical resources.

P depends on \mathcal{PR} for ω iff $\wedge \left\{ \begin{array}{l} \exists \mathcal{F} \subset \mathbb{F} : P \text{ depends on } \mathcal{F} \text{ for } \omega \\ \forall F \in \mathcal{F}, \exists PR \in \mathcal{PR} : \text{supports}(PR, F) \end{array} \right.$

For instance, John depends on the set {PC} for the activity “John edits toto.doc”.

The degree of a person’s dependence on a set of physical resources for an activity is given by the ratio of the ω -minimal paths that contain artifacts supported by this set of physical resources.

Definition 4.9 (Degree of a person’s dependence on a set of physical resources for an activity).

Let $\omega = (P, D, S)$ be an activity, let \mathcal{PR} be a set of physical resources, let \widehat{Y}^ω be the ω -minimal paths. The degree of dependence of P on \mathcal{PR} , noted $d_{\mathcal{PR}}^\omega$ is:

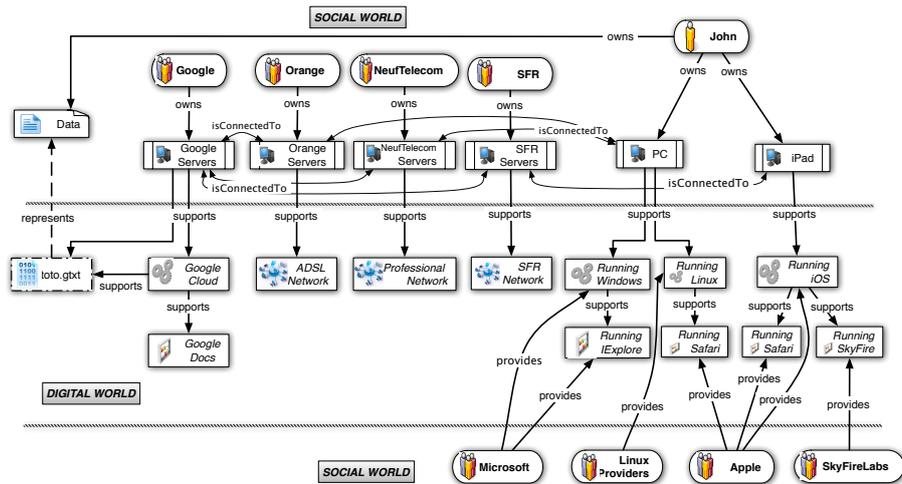
$$d_{\mathcal{PR}}^\omega = \frac{|\{\sigma : \sigma \in \widehat{Y}^\omega \wedge \exists PR \in \mathcal{PR}, \exists F \in \sigma, \text{supports}(PR, F)\}|}{|\widehat{Y}^\omega|}$$

For instance, the degree of dependence of John on the set {PC} for the activity “John edits the document toto.doc” is equal to 1.

By means of these definitions, we can be explicitly aware of the user’s dependences on the digital and social world. A use-case is presented in the next section to illustrate the previous notions.

4.5 USE-CASE EXAMPLE: GOOGLEDOCS

Figure 4.5 presents a simple system where a person uses GoogleDocs, drawn by applying SOCIOPATH. The objective of this use-case is to show the different facets that SOCIOPATH modeling can be used. In the social world, the person John owns some Data, a PC and an iPad. We explicitly name only some legal entities who provide resources and artifacts: Microsoft (providing Windows and Internet Explorer so called IExplorer), Google (providing GoogleDocs and Google Cloud

Figure 4.5 – *GoogleDocs snapshot*.

services), SkyFireLabs (providing the SkyFire application), Apple (providing the iOS operating system and the browser Safari) and Linux Providers (*cf.* Figure 4.3). NeufTelecom, Orange and SFR are telecom companies. John’s iPad is connected to SFR Servers and John’s PC is connected to NeufTelecom Servers and Orange Servers.

In the digital world, the operating systems Windows and Linux are running on John’s PC. Windows supports IExplorer and Linux supports Safari. John’s iPad supports the running iOS, which supports two applications, Safari and SkyFire. John’s data are represented in the digital world by the document `toto.gtxt` which is supported by the physical resources owned by Google. We consider Google Cloud as the storage system used by the application GoogleDocs.

Analysis and results: we analyze this example by using `SOCIOPATH` and show that this modeling can provide answers to the questions presented in the introduction of this chapter:

1. Who are the persons that have a possibility to access John’s data? and what are the potential coalitions among persons that could allow undesired access to this data?

By applying the deduction rules presented in Section 4.3, we deduce the relations of *accesses* and *controls* that exist in this architecture. They are illustrated in Figure 4.6.

By knowing the relations *accesses* in the architecture (*cf.* left side of Figure 4.6), John is able to know which persons have a possible path to his document. Thus, these persons can² access his data. In this example, the persons who have a possible path to John’s documents are: SFR, NeufTelecom, John, Orange, Google.

Furthermore, by examining the persons who control the artifacts in the paths (*cf.* right side of Figure 4.6), it is possible to understand

²By *can*, we mean that a user may be able to perform an action, and not that she is actually permitted to. In this work, we do not analyze yet access control and user permission constraints.

Group	Sets of persons John depends on	Group	Sets of persons John depends on
G1	{Microsoft}	G12	{Apple,Orange,NeufTelecom}
G2	{Linux Providers}	G13	{Microsoft,SkyFireLabs}
G3	{Apple}	G14	{Orange,SFR}
G4	{SkyFireLabs}	G15	{Apple,Orange}
G5	{SFR}	G16	{Microsoft,NeufTelecom}
G6	{NeufTelecom}	G17	{Microsoft,Orange}
G7	{Orange}	G18	{SkyFireLabs,NeufTelecom}
G8	{Google}	G19	{Microsoft,SFR,Linux Providers}
G9	{Microsoft,Apple}	G20	{Apple,NeufTelecom}
G10	{NeufTelecom,Orange,SFR}	G21	{Linux Providers,SkyFireLabs}
G11	{Linux Providers,SFR}		

Table 4.2 – Sets of persons John depends on.

By applying the definitions of Sections 4.4.2, we obtain John’s social and digital dependences, and the degree of these dependences for the activity “John reads `toto.gtxt`”. We show the results concerning some sets of persons John depends on in Table 4.2 and the degree of dependences on these sets in Figure 4.7. This information reveals how much John is autonomous from a specific person or a set of persons. For instance, the degree of dependence on `{Microsoft}` is 0.33, and the degree of dependence on the set `{Apple, NeufTelecom}` is 0.83.

- Who are the persons who can prevent John from performing the activity “John reads `toto.gtxt`”?

The list of sets on which the degree of dependence is equal to one are the persons who can prevent John from “reading `toto.gtxt`” because they cross all the ω -paths from John to his data. From Table 4.2, these sets are: $G8=\{\text{Google}\}$, $G9=\{\text{Microsoft}, \text{Apple}\}$, $G10=\{\text{NeufTelecom}, \text{Orange}, \text{SFR}\}$, $G12=\{\text{NeufTelecom}, \text{Orange}, \text{Apple}\}$, $G19=\{\text{Microsoft}, \text{SFR}, \text{Linux Providers}\}$.

- Who are the persons whom John is able avoid to perform the activity “John reads `toto.gtxt`”?

John depends on the sets on which the degree of dependence is less than one, in a less dramatic way (e.g., on the set `{SkyFireLabs, NeufTelecom}` with a degree of 0.5), because this shows that there are others minimal ω -paths enabling John to read `toto.gtxt` and the persons who belong to this set do not control any artifact in these paths. These sets enlighten the “combinations of persons” which John is able to avoid, at will.

SOCIOPATH could then be useful in the evaluation process of a system with respect to trust requirements. This leads to the fifth question presented in the introduction of this chapter:

- How much a user trusts a system for a specific activity?

We focus on this question in Chapter 5 and Chapter 6.

This use case scenario shows how, by applying SOCIOPATH on an architecture, a user can evaluate the system by taking into account the dependence-related aspects. In spite of the simplistic example proposed,

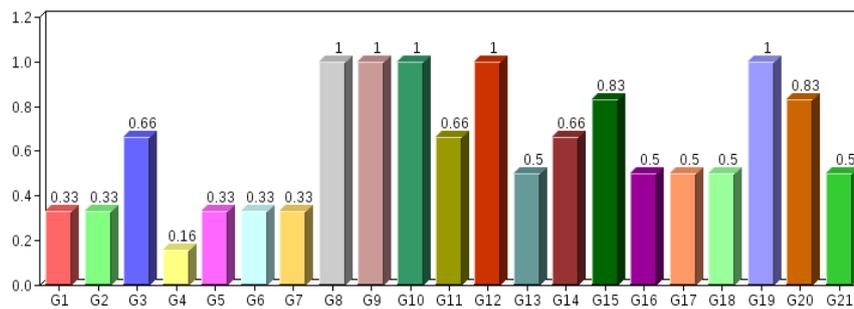


Figure 4.7 – Degree of dependence on persons' sets.

some of the outcomes go beyond the immediate understanding of a system an average user has. Thus SOCIOPATH exposes system's characteristics that we believe are worth evaluating.

After this detailed presentation of SOCIOPATH, we notice that what mainly distinguishes SOCIOPATH from EA tools is the analyzing of social dependences, more precisely, the sets of persons a user depends on to achieve an activity. In EA, a person is considered as an actor who plays a role to ensure the continuity of the business. The social aspects are limited to business point view like the enterprise's strategies, objectives and activities. Whereas, in SOCIOPATH, a person is represented as an entity in the social world who can be an actor that participates to the system and plays some role or can be a user who owns some data that allow her to achieve an activity. Hence, the relations among social participants to the system can be deduced.

4.6 CONCLUSION

In this chapter, we introduced SOCIOPATH, a meta-model that formalizes systems in order to reveal the relations of dependence among participants. A formalism of SOCIOPATH is given by several definitions, upon which we have defined some rules, based on first order logic. These rules only captures those aspects of the model that are needed to build the relations among the system's components. SOCIOPATH allows to deduce the dependences of a person on the entities of a system architecture for an activity like editing a document, sharing data, using a software, *etc.* With SOCIOPATH it is possible to know, whom can prevent a person to perform an activity or whom a person can avoid to perform an activity.

Currently SOCIOPATH does not distinguish the different kinds of access and control relations. In order to consider intentions and expectations of users regarding digital resources, SOCIOPATH can be enriched with access and control typologies, to define different kinds of dependences.

A SOCIOPATH model can be represented as a directed acyclic graph (DAG) as we show in next chapters. This DAG is used in our next contribution where we use mechanisms to evaluate user's trust in a system for an activity.

SOCIOTRUST: EVALUATING TRUST IN A SYSTEM FOR AN ACTIVITY USING THEORY OF PROBABILITY

“Probability is the very guide of
life.”

-Marcus Tullius Cicero.

CONTENTS

6.1	INTRODUCTION	67
6.2	INFERRING USER’S OPINION ON A SYSTEM USING SUBJECTIVE LOGIC	68
6.2.1	Opinion on a node for an activity	68
6.2.2	Opinion on a path for an activity	68
6.2.3	Opinion on a system for an activity	70
6.3	EXPERIMENTAL EVALUATION	73
6.3.1	Comparing the proposed methods	73
6.3.2	Studying the accuracy of the proposed methods	74
6.3.3	Social evaluation: a real case	76
6.4	CONCLUSION	78

THE second contribution of this thesis is *evaluating the user’s trust in a system for an activity*. We propose two different approaches which are based on SOCIOPATH models. In this chapter, we present the first one SOCIOTRUST.

As we show in Chapter 3, trust can be evaluated through a graph. In this Chapter, we simplify the representation of SOCIOPATH, to model an activity achieved through a system, by a directed acyclic graph (DAG) as shown in Section 5.2. Each node in the DAG is associated with a trust value so the DAG becomes a weighted directed acyclic graph (WDAG). Based on this WDAG, we use the rules of probability theory to evaluate the user trust in a system for an activity in Section 5.3. A solution for the problem of dependent paths presented in Section 3.3.2 is proposed as well. Section 5.4 presents the experiments that validate the proposed approach.

5.1 INTRODUCTION

Everyday, digital activities like chatting, mailing, blogging, buying online, and sharing data are achieved through systems composed of physical and digital resources (*e.g.*, servers, software components, networks and PCs). These resources are provided and controlled by persons (individual or legal entities) on whom we depend to execute these activities. The set of entities and the different relations between them form a complex system for a specific activity. When users need to choose a system to perform an activity, they are faced with a lot of available options. To choose a system, they evaluate it considering many criteria: functionality, ease of use, QoS, economical aspects, *etc.* Trust is also a key factor of choice. However, evaluating this trustworthiness is a problematic issue due to the system complexity.

As we show in Chapter 3, trust has been widely studied in several aspects of daily life. In the trust management community [Mar94, MFGL12, Vilo5, JIBo7, ZDB11, YHo7], two main issues arise: (i) *How to define the trust in an entity, knowing that entities can be persons, digital and physical resources?* (ii) *How to evaluate such value of trust in a system under a particular context?* This point embodies the main focus of this chapter.

We argue that studying trust in the separate entities that compose a system does not give a picture of how trustworthy a system is as a whole. Indeed, the trust in a system depends on its architecture, more precisely, on the way the implicit and explicit entities, which the users depends on to do their activities, are organized.

Trust can be evaluated through a graph (*cf.* Section 3.3). Inspired by this idea, we propose SOCIOTRUST [ASABL13], a graph-based trust approach to evaluate trust in a system for an activity. The system definition is based on SOCIOPATH (*cf.* Chapter 4) which allows to present the architecture of a system as a weighted directed acyclic graph as we show in next section. Levels of trust are then defined for each node in the graph according to the user who evaluates trust. By combining trust values using the theory of probability, we are able to estimate two different granularities of trust, namely, *trust in a path* and *trust in a system*, both for an activity to be performed by a person.

To evaluate our contribution, we conducted several experiments to analyze the impact of different characteristics of a system on the behavior of the obtained trust values. Experiments realized on both synthetic traces and real data sets allow us to validate the accuracy of our approach.

The main problem that faces trust evaluation is the existence of *dependent paths* (*cf.* Section 3.3.2). In our approach, this problem is resolved using conditional probability.

Next section shows how to present SOCIOPATH model as a directed acyclic graph to be used for evaluating trust.

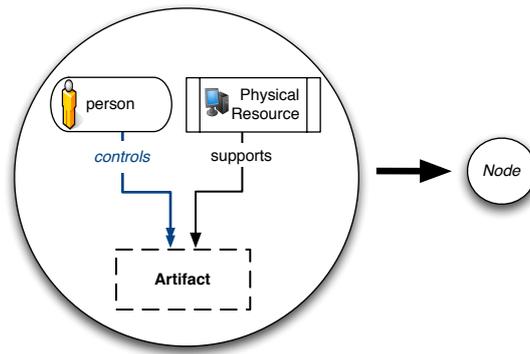


Figure 5.1 – The node of the WDAG.

5.2 A SOCIOPATH MODEL AS A DIRECTED ACYCLIC GRAPH (DAG)

We simplify the representation of SocioPath models by combining one artifact, the set of persons controlling it and the set of physical resources supporting it into one unique component. These merged components are the nodes in the DAG. The edges in the DAG represent the relations *access*. A user performs an activity by passing through successive *access* relations of the graph, so-called an *activity minimal path*¹.

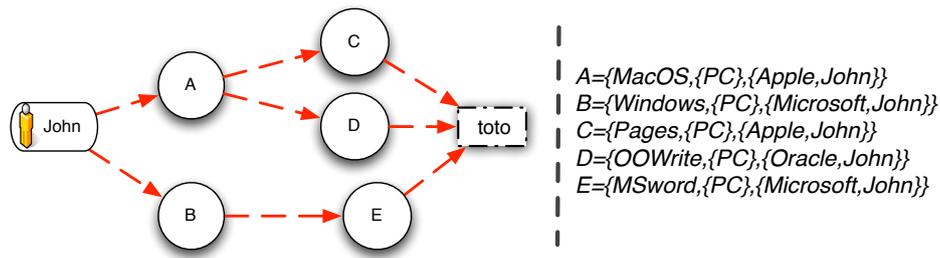
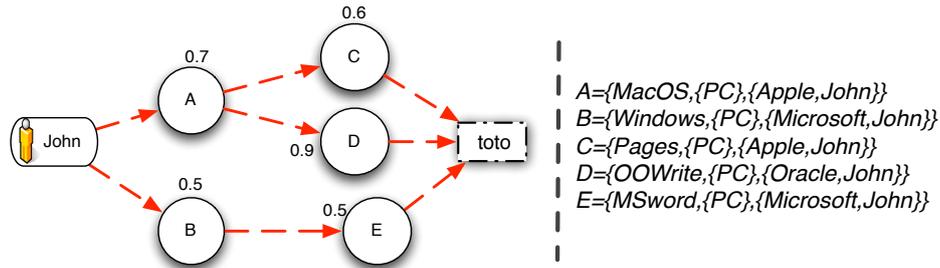
Definition 5.1 (A system for an activity).

A system that enables a user to achieve an activity can be formally expressed as a tuple: $\alpha = \langle \mathbb{N}_\omega, \mathbb{A}_\omega \rangle$ where:

- ω represents the activity the user wants to achieve.
- \mathbb{N}_ω represents the set of nodes in a system for an activity. Each node aggregates one artifact, the persons who control it and the physical resources that support it as shown in Figure 5.1.
- $\mathbb{A}_\omega \subseteq \mathbb{N}_\omega \times \mathbb{N}_\omega$ represents the set of edges in a system. From the rules of SocioPath and the aggregation we made for a node, our DAG exhibits only the relation *access*.

Figure 5.2 shows the same system presented in Figure 4.2 (page 38) as a merged DAG where each node represents an artifact with all additional information as physical resources that support it and persons who control it. For instance, the node *A* in Figure 5.2 represents the artifact MacOS with the additional information: {Apple, John} as the set of persons who control it and {PC} as the set of physical resource which support it. Each edge represents the relation *accesses*. The paths that enable John to access toto become: $\sigma_1 = \{A, C\}$; $\sigma_2 = \{A, D\}$; $\sigma_3 = \{B, E\}$ as shown in Figure 5.2.

¹If there is no ambiguity, we denote an activity minimal path through the DAG simply by a path σ and we do not consider the source and the target node.

Figure 5.2 – The activity “John accesses a document *toto*” as a DAG.Figure 5.3 – The activity “John accesses a document *toto*” as a WDAG.

5.3 A PROBABILISTIC APPROACH TO INFER SYSTEM TRUST VALUE

If a user needs to evaluate her trust in a system for an activity, she associates each node in the DAG with a trust value and the DAG becomes a weighted directed acyclic graph (WDAG). The notations used here are summarized in Table 5.1.

We define a function that associates each node with a trust value as follows.

$t : \mathbb{N} \rightarrow [0, 1]$ represents a function that assigns to each node a person’s trust level, which we assume to be within the interval $[0, 1]$, where 0 means not trustworthy at all and 1 means fully trustworthy.

The associated value on the node in Figure 5.3 represents the level of John’s trust in this node.

In this thesis, we adopt the definition of Jøsang about trust [JIB07] “trust is the probability by which an individual, *A*, expects that another individual, *B*, performs a given action on which its welfare depends” (cf. Section 3.2.1).

According to the previous definition, we consider the following.

- **Trust in a node for an activity:** The trust value of a user *P* in a node *N* for an activity ω is the probability, by which *P* believes that *N* provides her the expected services for ω . Then, we have $t(N) = \mathbf{P}(\lambda^N)$.
- **Trust in a path for an activity:** The trust value of a user *P* in a path σ for an activity ω is the probability, by which *P* believes that σ enables her to achieve ω . Then, we have $t(\sigma) = \mathbf{P}(\lambda^\sigma)$.
- **Trust in a system for an activity:** The trust value of a user *P* in a

Concept	Notation	Remark
Node	N	For a given activity ω achieved by a person P , the symbols ω and P are omitted in these notations for simplicity.
Trust in a node for an activity	$t(N)$	
Trust in a path for an activity	$t(\sigma)$	
Trust in a system for an activity	$t(\alpha)$	
The event “ N provides the expected services” for an activity”	λ^N	
The event “ P achieves an activity through the path σ ”	λ^σ	
The event “ P achieves an activity through the system”	λ^α	
The probability of an event	$\mathbf{P}(\lambda)$	

Table 5.1 – Glossary of notations (2).

system α for an activity ω is the probability, by which P believes that α enables her to achieve ω . Then, we have $t(\alpha) = \mathbf{P}(\lambda^\alpha)$.

In the next sections, we formalize the previous definitions.

5.3.1 Trust in a node for an activity

Trust in a node is evaluated from the point of view of the concerned user. There are several ways to construct this trust level. We can figure out different objective and subjective factors that impact this trust level like the reputation of the persons who control the artifact, their skills, the performance of the physical resource that supports the artifact or the personal experience with this artifact. We thus have $t(N) = f(t_\omega^F, t_\omega^P, t_\omega^{PR})$, where $t_\omega^F, t_\omega^P, t_\omega^{PR}$ are respectively the trust value assigned to an artifact F , the set of persons \mathcal{P} who control F , the set of physical resources \mathcal{PR} which support F for a given activity ω . The meaning of the resulting trust value in a node depends on the employed function f to compute this value [MC96]. For instance, if Bayesian inference is employed to evaluate it as is done in [LW10], the node trust value is considered as “the probability by which a user believes that a node can perform an expected action for a given activity” [Gamoo]. However, in this thesis, we do not address the issue of computing the trust value of a user in a node for an activity but we interpret it as the probability, by which a user P believes that a node N provides her the expected services for ω . Then, we have

$$t(N) = \mathbf{P}(\lambda^N) \quad (5.1)$$

5.3.2 Trust in a path for an activity

A path in a system represents a way to achieve an activity. The trust level of a person P to achieve an activity through a particular path $\sigma = \{N_1, N_2, \dots, N_n\}$ is the probability that all the nodes $\{N_i\}_{i \in [1..n]}$ provides the expected services for the activity. Thus $\mathbf{P}(\lambda^\sigma)$ is computed as follows:

$$t(\sigma) = \mathbf{P}(\lambda^\sigma) = \mathbf{P}(\lambda^{N_1} \wedge \lambda^{N_2} \wedge \dots \wedge \lambda^{N_n})$$

The event λ^{N_i} means that N_i provides the expected services for an activity. Since the graph is acyclic (only minimum activity paths are considered), then the nodes N_1, \dots, N_n are different in the path, thus each λ^{N_i} is independent from all others. Hence, we can rewrite the trust in a path as follows:

$$t(\sigma) = \mathbf{P}(\lambda^\sigma) = \mathbf{P}(\lambda^{N_1}) \times \mathbf{P}(\lambda^{N_2}) \times \dots \times \mathbf{P}(\lambda^{N_n}) = \prod_{i=1}^n \mathbf{P}(\lambda^{N_i}) \quad (5.2)$$

5.3.3 Trust in a system for an activity

A system, which is often composed of several paths, represents the several ways a person has, to achieve an activity. The trust level of a person P in a system α to achieve an activity is the probability that she achieves her activity through one of the paths in the system. To evaluate the trust in a system for an activity, two cases have to be considered: (i) the paths are independent *i.e.*, they have no nodes in common² and (ii) the paths are dependent *i.e.*, there exists at least one node in common.

A node contains an artifact with some additional information like the persons who control the artifact and the physical resources that support it (*cf.* Section 5.2). For reasons of simplicity, we consider that the independent paths are the paths that have no nodes in common without discussing the cases where the nodes share only one component like a person who controls the artifact.

5.3.3.1 Independent paths

Let $\{\sigma_i\}_{i \in [1..m]}$ be independent paths that enable a person P to achieve an activity. The probability of achieving the activity through a system, $\mathbf{P}(\lambda^\alpha)$, is the probability of achieving the activity through one of the paths σ_i . Thus $\mathbf{P}(\lambda^\alpha)$ is computed as follows:

$$t(\alpha) = \mathbf{P}(\lambda^\alpha) = \mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \dots \vee \lambda^{\sigma_m})$$

Since the paths are independent then the equation can be rewritten as follows:

$$t(\alpha) = \mathbf{P}(\lambda^\alpha) = 1 - \prod_{i=1}^m (1 - \mathbf{P}(\lambda^{\sigma_i})) \quad (5.3)$$

For instance, if a person has two independent paths to achieve an activity then:

$$\begin{aligned} t(\alpha) &= \mathbf{P}(\lambda^\alpha) = \mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2}) \\ &= 1 - (1 - \mathbf{P}(\lambda^{\sigma_1})) \times (1 - \mathbf{P}(\lambda^{\sigma_2})) \\ &= \mathbf{P}(\lambda^{\sigma_1}) + \mathbf{P}(\lambda^{\sigma_2}) - \mathbf{P}(\lambda^{\sigma_1}) \times \mathbf{P}(\lambda^{\sigma_2}) \end{aligned} \quad (5.4)$$

5.3.3.2 Dependent paths

When there are common nodes between paths, the Relation 5.3 can not be applied directly (*cf.* Section 3.3.2). To evaluate the trust through dependent paths, we begin from a simple case where a system has two paths before generalizing.

²The dependent paths in our graph are the paths that have common nodes and not common edges as in social networks because the trust value is associated to a node and not to an edge as in a social network.

1. **Two dependent paths with one common node:** Let σ_1, σ_2 , be two paths that enable a person P to achieve an activity. $\sigma_1 = \{N, N_{1,2}, \dots, N_{1,n}\}$, $\sigma_2 = \{N, N_{2,2}, \dots, N_{2,m}\}$. These two paths have a common node, which is N so they are dependent. Thus the probability that a person P achieves the activity ω through the system α is computed as follows:

$$t(\alpha) = \mathbf{P}(\lambda^\alpha) = \mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2}) = \mathbf{P}(\lambda^{\sigma_1}) + \mathbf{P}(\lambda^{\sigma_2}) - \mathbf{P}(\lambda^{\sigma_1} \wedge \lambda^{\sigma_2})$$

The probability $\mathbf{P}(\lambda^{\sigma_1} \wedge \lambda^{\sigma_2})$ can be rewritten using conditional probability as the two paths are dependent.

$$\begin{aligned} t(\alpha) &= \mathbf{P}(\lambda^\alpha) = \mathbf{P}(\lambda^{\sigma_1}) + \mathbf{P}(\lambda^{\sigma_2}) - \mathbf{P}(\lambda^{\sigma_2}) \times \mathbf{P}(\lambda^{\sigma_1} | \lambda^{\sigma_2}) \\ &= \mathbf{P}(\lambda^{\sigma_1}) + \mathbf{P}(\lambda^{\sigma_2}) \times (1 - \mathbf{P}(\lambda^{\sigma_1} | \lambda^{\sigma_2})) \end{aligned}$$

We have to compute $\mathbf{P}(\lambda^{\sigma_1} | \lambda^{\sigma_2})$ which is the probability that P achieves the activity through σ_1 once it is already known that P achieves the activity through σ_2 . Thus it is the probability that $N, \{N_{1,i}\}_{i \in [2..n]}$ provides the expected services for this activity, once it is known that $N, \{N_{2,i}\}_{i \in [2..m]}$ provided the expected services. Thus N has already provided the expected services. Hence, $\mathbf{P}(\lambda^{\sigma_1} | \lambda^{\sigma_2}) = \prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}})$, where $\lambda^{N_{1,i}}$ is the event “ $N_{1,i}$ provides the necessary services for the activity”.

$$\begin{aligned} t(\alpha) &= \mathbf{P}(\lambda^\alpha) \\ &= \mathbf{P}(\lambda^N) \times \prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}}) + \mathbf{P}(\lambda^N) \times \prod_{i=2}^m \mathbf{P}(\lambda^{N_{2,i}}) \times (1 - \prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}})) \\ &= \mathbf{P}(\lambda^N) \times \left[\prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}}) + \prod_{i=2}^m \mathbf{P}(\lambda^{N_{2,i}}) \times (1 - \prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}})) \right] \\ &= \mathbf{P}(\lambda^N) \times \left[\prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}}) + \prod_{i=2}^m \mathbf{P}(\lambda^{N_{2,i}}) - \prod_{i=2}^m \mathbf{P}(\lambda^{N_{2,i}}) \times \prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}}) \right] \end{aligned}$$

From Equation 5.4 we can note that the term:

$$\prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}}) + \prod_{i=2}^m \mathbf{P}(\lambda^{N_{2,i}}) - \prod_{i=2}^m \mathbf{P}(\lambda^{N_{2,i}}) \times \prod_{i=2}^n \mathbf{P}(\lambda^{N_{1,i}})$$

is the probability that P achieves the activity through $\sigma'_1 = \{N_{1,2}, \dots, N_{1,n}\}$ or $\sigma'_2 = \{N_{2,2}, \dots, N_{2,m}\}$ which are the paths after eliminating the common nodes. Thus the previous equation can be rewritten as follows:

$$t(\alpha) = \mathbf{P}(\lambda^\alpha) = \mathbf{P}(\lambda^N) \times \mathbf{P}(\lambda^{\sigma'_1} \vee \lambda^{\sigma'_2})$$

2. **Two dependent paths with several common nodes:** Let σ_1, σ_2 , be two paths that enable a person P to achieve an activity. These two paths have several common nodes. By following the same logic as

in 5.3.3.2.1, we compute the probability that a person P achieves activity ω through system α as follows:

$$t(\alpha) = \mathbf{P}(\lambda^\alpha) = \prod_{N \in \sigma_1 \cap \sigma_2} \mathbf{P}(\lambda^N) \times \mathbf{P}(\lambda^{\sigma'_1} \vee \lambda^{\sigma'_2})$$

where $\sigma'_1 = \sigma_1 \setminus \sigma_2$, $\sigma'_2 = \sigma_2 \setminus \sigma_1$.

3. **Several dependent paths:** A person may have several paths l with common nodes. Thus $\mathbf{P}(\lambda^\alpha)$ is computed as follows:

$$\begin{aligned} t(\alpha) = \mathbf{P}(\lambda^\alpha) &= \mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \dots \vee \lambda^{\sigma_l}) = \\ &\mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \dots \vee \lambda^{\sigma_{l-1}}) + \mathbf{P}(\lambda^{\sigma_l}) - \mathbf{P}(\lambda^{\sigma_l}) \times \mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \dots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l}) \end{aligned} \quad (5.5)$$

Let us discuss these terms one by one:

- The term $\mathbf{P}(\lambda^{\sigma_l})$ can be computed directly from Equation 5.2.
- The term $\mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \dots \vee \lambda^{\sigma_{l-1}})$ can be computed recursively using Equation 5.5.
- The term $\mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \dots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l})$ needs first to be simplified. If we follow the same logic we discussed in Section 5.3.3.2.1, the term $\mathbf{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \dots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l})$ can be replaced by the term $\mathbf{P}(\lambda^{\sigma'_1} \vee \lambda^{\sigma'_2} \vee \dots \vee \lambda^{\sigma'_{l-1}})$ where we obtain each $\lambda^{\sigma'_i}$ by eliminating the nodes in common with σ_l .
- $\mathbf{P}(\lambda^{\sigma'_1} \vee \lambda^{\sigma'_2} \vee \dots \vee \lambda^{\sigma'_{l-1}})$ can be computed recursively using Equation 5.5, and recursion is guaranteed to terminate while the number of paths is finite.

We are now able to evaluate the trust in a whole system α .

5.4 EXPERIMENTAL EVALUATIONS

In this section, we present different experiments, their results, analysis and interpretation. The main objectives for conducting the experiments are: (i) to study the influence of the system organization on the computed trust values and (ii) to confront this approach with real users. The first two experiments are related to the first objective while the third experiment is devoted to the second.

5.4.1 Influence of the system architecture on the trust value

The objective of this experiment is to study the influence of the system organization on the computed trust value. We apply our equations on different systems that have the same number of nodes A, B, C, D, E, F and the same values of trust assigned to each node, but assembled in different topologies as presented in Table 5.2. The values of trust associated to nodes A, B, C, D, E, F are 0.1, 0.2, 0.3, 0.9, 0.8, 0.7 respectively.

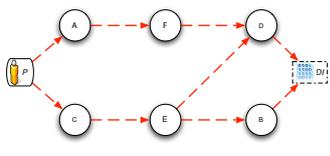
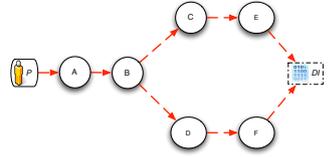
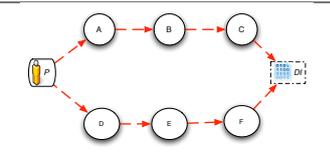
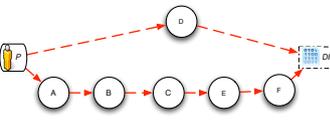
	α	$t_{\omega}(\alpha)$
α_1		0.4409
α_2		0.0144
α_3		0.507
α_4		0.9003

Table 5.2 – Different systems and their trust value.

We compute the trust value $t(\alpha)$ of each system. We obtain very divergent results varying from 0.0144 to 0.9003 as illustrated in Table 5.2. Collecting the values of trust in each separated node in a system is not enough to determine if the system is trustworthy or not for an activity. One must also know how the system is organized. For example, in α_2 , all the paths contain the nodes A and B and the trust values in these nodes is quite low, 0.1 and 0.2 respectively, so the system trust value is also low due to the strong dependency on these two nodes in this system.

5.4.2 Influence of the path length and the number of paths on the trust value

We conducted several simulations to observe the evolution of the trust value for an activity according to some characteristics in the graph. As a dataset, we considered random graphs composed of 20 to 100 nodes, and of 1 to 15 paths. Each node in the graph is associated with a random value of trust from a predefined range.

Firstly, the evolution of trust values according to the path lengths in a graph is evaluated. Each simulated graph is composed of 5 paths with lengths varying from 1 to 15 nodes. Different ranges of trust values towards the nodes were simulated, namely: $[0.1, 0.4]$, $[0.4, 0.6]$, $[0.6, 0.9]$ and $[0.1, 0.9]$. Figure 5.4 illustrates the impact of the path lengths on the trust value. Note that, the system trust value decreases when the length of paths increases. This reflects the natural intuition that the measure of trust in a path falls as the path gets longer.

Secondly, we set the path lengths to 5 nodes and we increased the number of paths from 1 up to 15 in order to observe the variation of the trust values. Again, different node trust values were simulated: $[0.1, 0.3]$,

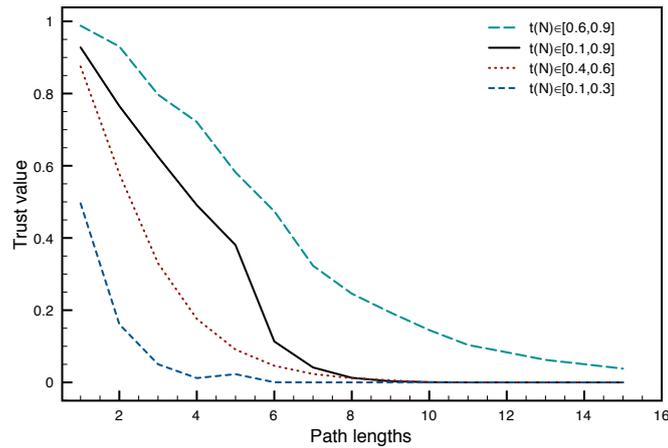


Figure 5.4 – System trust value according to the length of paths.

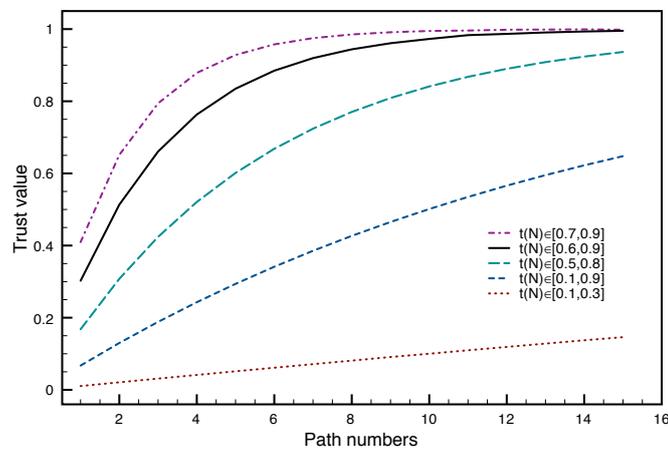


Figure 5.5 – System trust value according to the number of paths.

$[0.5,0.8]$, $[0.6,0.9]$, $[0.7,0.9]$ and $[0.1,0.9]$. Simulation results are reported in Figure 5.5 which shows that the trust value increases as the number of paths increase. This reflects the intuition that the measure of trust in a system for an activity rises when the number of ways to achieve this activity increases.

5.4.3 Social evaluation: a real case

In order to evaluate our proposal in a real use case, we modeled a subpart of the LINA research laboratory system³ using SOCIOPATH. We applied the rules of SOCIOPATH on this system for the activity “a user accesses a document `todo` that is stored on the SVN server at LINA”. Due to privacy issues, Figure 5.6 presents only the DAG of LINA architecture for this activity, with anonymous nodes. For the sake of clarity, we simplify the underlying graph as much as possible.

Based on this context, we conducted an opinion survey among twenty members of LINA including, PhD students, professors and computer technicians about their level of trust in each node. For each person, we have

³<https://www.lina.univ-nantes.fr/>

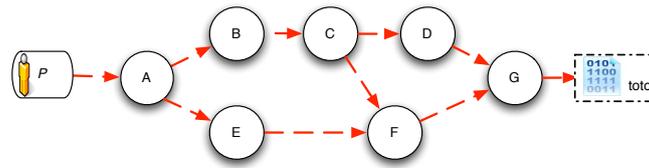


Figure 5.6 – LINA’s WDAG for the activity “accessing a document *toto* on the SVN”.

	A	B	C	D	E	F	G	System trust value	User’s feedback
P_1	0.5	0.5	1	0.5	0.5	1	1	0.4375	✓
P_2	0.7	1	1	0.7	0.7	1	1	0.847	✓
P_3	0.5	0.5	1	0.7	0.5	1	1	0.4375	×
P_4	0.6	0.6	0.8	0.7	0.6	0.8	0.6	0.3072	×
P_5	0.8	0.8	1	0.8	0.8	1	0.9	0.8202	✓
P_6	0.9	0.9	1	0.9	0.9	0.9	0.9	0.9043	✓
P_7	0.6	0.6	0.7	0.6	0.6	0.6	0.7	0.2770	×
P_8	0.8	0.6	1	0.9	0.8	0.8	1	0.7416	✓
P_9	0.7	0.5	1	0.4	0.7	0.6	0.9	0.4407	✓
P_{10}	0.8	1	0.7	0.8	0.8	0.9	0.8	0.6975	✓
P_{11}	0.5	0.5	0.9	0.5	0.5	0.5	0.9	0.2473	×
P_{12}	0.95	0.95	0.8	0.8	0.95	0.95	0.8	0.8655	✓
P_{13}	0.8	0.9	0.8	0.7	0.95	0.8	0.7	0.6433	✓
P_{14}	0.8	0.7	0.9	0.7	0.9	0.8	0.8	0.6652	✓
P_{15}	0.9	0.8	0.8	0.9	0.9	0.9	0.8	0.7733	✓
P_{16}	0.7	0.6	0.6	0.6	0.8	0.7	0.6	0.337	✓
P_{17}	0.5	0.9	0.8	0.7	0.9	0.5	0.8	0.3807	×
P_{18}	0.7	0.7	1	0.7	0.6	0.7	1	0.6088	✓
P_{19}	0.8	0.8	1	1	1	0.8	1	0.8704	✓
P_{20}	0.9	0.9	0.8	0.9	0.9	0.9	0.8	0.7971	✓

Table 5.3 – User’s trust value in the system SVN in LINA.

computed the system trust value according to the methodology presented in Section 5.3. Table 5.3 presents The survey data and the computed trust value in the system according to LINA members. Over a second phase, we asked each user for feedback about the system trust values computed with respect to their level of trust in the nodes. The last column of Table 5.3 shows this feedback, where ✓ means that they are satisfied with the value, and × means that they are not satisfied. 75% of the users are satisfied with the computation. Unsatisfied users argue that they expect a higher trust value. Some of trust values associated to the nodes of the unsatisfied users, have relatively low values (around 0.5 or 0.6) compared to the other users. These users explain that the lack of knowledge about some nodes leads them to vote with a neutral value (0.5 or 0.6) which for them considered neither trustworthy, nor untrustworthy. Clearly, such behavior is not compatible with a probabilistic interpretation where 0.5 is no more *neutral* than any other possible value. The explanations provided by users reveal an interesting point; even in the case of a local environment and even considering advanced users, not everyone is in possession of all the information necessary for an informed assessment. To conform to this reality and model this phenomenon, it requires to use a formalism allowing to express uncertainty related to incompleteness of available information. Classical probability theory is limited in expressing ignorance or uncertainty while subjective logic [Jøs01], which is an extension of the probability theory, can deal with these issues.

5.5 CONCLUSION

In this chapter, we proposed `SocioTrust`, a probabilistic approach to evaluate the system trust for an activity. We conducted some experiments to illustrate that the system construction is a key factor in evaluating the user trust value in a system. Finally, we confronted our approach with real user opinions based on a real modeled system to extract the limitations of this proposition.

In our study, trust value has been considered as a traditional probability without any degree of uncertainty. The experiment in Section 5.4.3 shows that in real situation, not everyone is in possession of all the necessary information to provide a dogmatic opinion about something or someone. Extending our approach to use subjective logic, which can express uncertainty, is the objectif of Chapter 6.

SUBJECTIVETRUST: EVALUATING TRUST IN A SYSTEM FOR AN ACTIVITY USING SUBJECTIVE LOGIC

“As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.”

-Albert Einstein.

CONTENTS

IN this chapter, we extend the proposition of **SOCIO**TRUST to be able to deal with uncertainty in the trust values. We propose **SUBJECTIVE**TRUST. Our approach is based on subjective logic, an extension of probabilistic logic that takes uncertainty into account.

Inspired by the idea of modeling trust using a graph, we simplify the representation of **SOCIO**PATH to model, an activity achieved through a system, by a directed acyclic graph (DAG) as we do in **SOCIO**TRUST. Each node in the DAG is associated with an *opinion* so the DAG becomes a **WDAG**. Based on this **WDAG**, we use subjective logic to evaluate the user trust in a system for an activity in Section 6.2. Some solutions for the problem of dependent paths are proposed as well. Section 6.3 presents the conducted experiments for evaluating this proposal.

6.1 INTRODUCTION

Previous chapter introduces SOCIOTRUST. It is a graph-based trust approach based on probability theory, to evaluate trust in a system for an activity. SOCIOTRUST is an approach that works perfectly in full-knowledge environments. However, in uncertain environments, users might not be in possession of all the information to provide a dogmatic opinion and traditional probability cannot express uncertainty.

With subjective logic [Jøs01], trust can be expressed as subjective opinions with degrees of uncertainty. In this chapter, we aim to take advantage of the benefits of subjective logic to evaluate trust.

As we said in Section 3.2.2.2, in the terminology of subjective logic, an opinion held by an individual P about a proposition x is the ordered quadruple $O_x = (b_x, d_x, u_x, a_x)$ where:

- b_x (belief) is the belief that x is true.
- d_x (disbelief) is the belief that the x is false.
- u_x (uncertainty) is the amount of uncommitted belief.
- a_x is called the base rate, it is the a priori probability in the absence of evidence.

Note that $b_x, d_x, u_x, a_x \in [0..1]$ and $b_x + d_x + u_x = 1$. a_x is used for computing an opinion's probability expectation value that can be determined as $\mathbb{E}(O_x) = b_x + a_x u_x$. More precisely, a_x determines how uncertainty shall contribute to the probability expectation value $\mathbb{E}(O_x)$ [Jøs01].

In this chapter, we extend SOCIOTRUST to use subjective logic. The main contribution of this chapter is proposing a generic model named SUBJECTIVETRUST [ABSAL13], for evaluating trust in a system for an activity taking into account uncertainty.

The system definition is also based on SOCIOPATH (cf. Chapter 4). To focus on the trust in the system, the SOCIOPATH model is abstracted in a DAG as in SOCIOTRUST (cf. Section 5.2). In subjective logic trust is expressed as an opinion, thus in this proposition, the DAG is weighted with opinions *i.e.*, each node is associated with an opinion. The function that assigns each node with an opinion is defined as follows:

$$O : \mathbb{N} \rightarrow ([0, 1], [0, 1], [0, 1], [0, 1]).$$

Figure 6.1 shows the same system presented in Figure 4.2 as a merged WDAG. The associated value on the node represents John's opinion on this node.

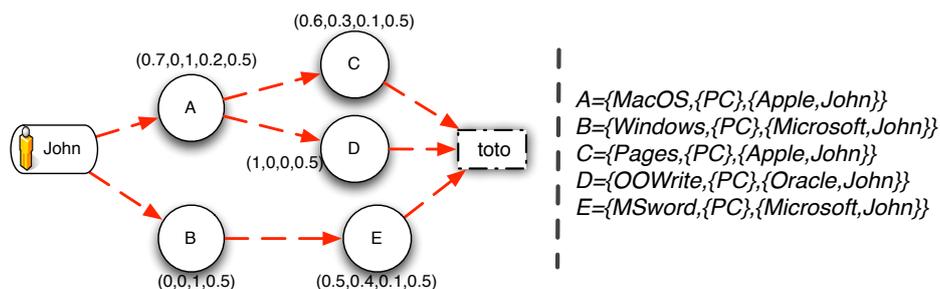


Figure 6.1 – The activity “John accesses a document *toto*” as a WDAG where the weights are opinions.

By combining the user's opinion on a node, we are able to estimate two different granularities of trust, namely, *opinion on a path* and *opinion on a system*, both for an activity to be performed by a person.

The main problem that faces trust evaluation based on a graph is the existence of *dependent paths* (cf. Section 3.3.2). To solve this problem, we propose two methods, **Copy** and **Split**. Next section shows how to use subjective logic in our approach.

6.2 INFERRING USER'S OPINION ON A SYSTEM USING SUBJECTIVE LOGIC

This section presents SUBJECTIVETRUST, a graph-based trust approach to infer trust in a system for an activity using subjective logic. User's opinions are associated to the nodes in the graph. Then, an opinion on a path and an opinion on a system are evaluated by combining respectively the opinions on the nodes and the opinions on the paths, using the appropriate operators of subjective logic. We propose two methods to solve the problem of dependent paths.

6.2.1 Opinion on a node for an activity

As in SOCIOTRUST, opinion on a node is evaluated from the point of view of the concerned user depending on her personal experience with this node. Several approaches have been proposed to obtain this opinion. In [Jøso1], authors translate the user's negative or positive observations to opinions. In [LW10, LW11b], the opinion parameters are estimated by Bayesian inference.

However, in this thesis, we do not address the issue of obtaining this opinion, we focus on combining the opinions associated on the nodes to obtain an opinion on a path and on a system for an activity.

6.2.2 Opinion on a path for an activity

Subjective logic consists of a set of logical operations like conjunction, disjunction, consensus and discounting operators which are defined to combine opinions. In the following, we present these operators [Jøso1].

- Conjunction represents the opinion of a person on several propositions. Let $O_x^P = (b_x^P, d_x^P, u_x^P, a_x^P)$ and $O_y^P = (b_y^P, d_y^P, u_y^P, a_y^P)$ be respectively P 's opinion on x and y . $O_{x \wedge y}^P$ represents P 's opinion on both x and y and can be calculated with the following relations:

$$O_x^P \wedge O_y^P = O_{x \wedge y}^P = \begin{cases} b_{x \wedge y}^P = b_x^P b_y^P \\ d_{x \wedge y}^P = d_x^P + d_y^P - d_x^P d_y^P \\ u_{x \wedge y}^P = b_x^P u_y^P + u_x^P b_y^P + u_x^P u_y^P \\ a_{x \wedge y}^P = \frac{b_x^P u_y^P a_y^P + b_y^P u_x^P a_x^P + u_x^P a_x^P u_y^P a_y^P}{b_x^P u_y^P + u_x^P b_y^P + u_x^P u_y^P} \end{cases} \quad (6.1)$$

$$\mathbb{E}(O_x^P \wedge O_y^P) = \mathbb{E}(O_{x \wedge y}^P) = \mathbb{E}(O_x^P) \mathbb{E}(O_y^P) \quad (6.2)$$

- Disjunction represents the opinion of a person on one of the propositions or any union of them. Let $O_x^P = (b_x^P, d_x^P, u_x^P, a_x^P)$ and $O_y^P = (b_y^P, d_y^P, u_y^P, a_y^P)$ be respectively P 's opinion on x and y . $O_{x \vee y}^P$ represents P 's opinion on x or y or both and can be calculated with the following relations:

$$O_x^P \vee O_y^P = O_{x \vee y}^P = \begin{cases} b_{x \vee y}^P = b_x^P + b_y^P - b_x^P b_y^P \\ d_{x \vee y}^P = d_x^P d_y^P \\ u_{x \vee y}^P = d_x^P u_y^P + u_x^P d_y^P + u_x^P u_y^P \\ a_{x \vee y}^P = \frac{u_x^P a_x^P + u_y^P a_y^P - b_x^P u_y^P a_y^P - b_y^P u_x^P a_x^P - u_x^P a_x^P u_y^P a_y^P}{u_x^P + u_y^P - b_x^P u_y^P - b_y^P u_x^P - u_x^P u_y^P} \end{cases} \quad (6.3)$$

$$\mathbb{E}(O_x^P \vee O_y^P) = \mathbb{E}(O_{x \vee y}^P) = \mathbb{E}(O_x^P) + \mathbb{E}(O_y^P) - \mathbb{E}(O_x^P)\mathbb{E}(O_y^P) \quad (6.4)$$

- Discounting represents the transitivity of the opinions. Let $O_B^P = (b_B^P, d_B^P, u_B^P, a_B^P)$ be P 's opinion on B 's advice, and $O_x^B = (b_x^B, d_x^B, u_x^B, a_x^B)$ be B 's opinion on x , $O_x^{PB} = O_B^P \otimes O_x^B$ represents P 's opinion on x as a result of B 's advice to P :

$$O_x^{PB} = O_B^P \otimes O_x^B = \begin{cases} b_x^{PB} = b_B^P b_x^B \\ d_x^{PB} = b_B^P d_x^B \\ u_x^{PB} = d_B^P + u_B^P + b_B^P u_x^B \\ a_x^{PB} = a_x^B \end{cases} \quad (6.5)$$

- Consensus represents the consensus of the opinions of different persons. Let $O_x^A = (b_x^A, d_x^A, u_x^A, a_x^A)$ be A 's opinion on x , and $O_x^B = (b_x^B, d_x^B, u_x^B, a_x^B)$ be B 's opinion on x , $O_x^{A,B} = O_x^A \oplus O_x^B$ represents the opinion of the group of persons $\{A, B\}$ on x :

$$O_x^{A,B} = O_x^A \oplus O_x^B = \begin{cases} b_x^{A,B} = \frac{b_x^A u_x^B + b_x^B u_x^A}{u_x^A + u_x^B - u_x^A u_x^B} \\ d_x^{A,B} = \frac{d_x^A u_x^B + d_x^B u_x^A}{u_x^A + u_x^B - u_x^A u_x^B} \\ u_x^{A,B} = \frac{u_x^A u_x^B}{u_x^A + u_x^B - u_x^A u_x^B} \\ a_x^{A,B} = \frac{u_x^A a_x^B + u_x^B a_x^A - (a_x^A + a_x^B) u_x^A u_x^B}{u_x^A + u_x^B - 2u_x^A u_x^B} \end{cases} \quad (6.6)$$

It is important to mention that conjunction and disjunction are commutative and associative.

$$O_x^P \wedge O_y^P = O_y^P \wedge O_x^P$$

$$O_x^P \vee O_y^P = O_y^P \vee O_x^P$$

$$(O_x^P \wedge O_y^P) \wedge O_z^P = O_x^P \wedge (O_y^P \wedge O_z^P)$$

$$(O_x^P \vee O_y^P) \vee O_z^P = O_x^P \vee (O_y^P \vee O_z^P)$$

However, the conjunction over the disjunction is not distributive. This is due to the fact that opinions must be assumed to be independent, whereas distribution always introduces an element of dependence.

$$O_x^P \wedge (O_y^P \vee O_z^P) \neq (O_x^P \wedge O_y^P) \vee (O_x^P \wedge O_z^P)$$

For the same reason, the discounting over the consensus is not distributive.

$$O_x^P \otimes (O_y^P \oplus O_z^P) \neq (O_x^P \otimes O_y^P) \oplus (O_x^P \otimes O_z^P)$$

By using these operators, we combine the opinions on the nodes to estimate two different granularities of trust: opinion on a path and opinion on a system.

A path in a system represents a way to achieve an activity. An opinion on a path that contains several nodes can be computed by combining the opinions on the nodes that belong to it.

In trust propagation, the used operator to build an opinion on a path is the discounting operator because it allows to compute the transitivity of an opinion along a path [JHPo6, JBo8].

However, if a person needs to achieve an activity through a path, she needs to pass by all the nodes composing this path. Hence, an opinion on a path is the opinion on all nodes composing this path.

The conjunction operator represents the opinion of a person on several propositions. If $O_x^P = (b_x^P, d_x^P, u_x^P, a_x^P)$ is P 's opinion on x and $O_y^P = (b_y^P, d_y^P, u_y^P, a_y^P)$ is P 's opinion on y , $O_{x \wedge y}^P$ represents P 's opinion on both x and y . Thus, the conjunction operator is the appropriate operator to compute an opinion on a path from the opinions on the nodes.

Let $\sigma = \{N_1, N_2, \dots, N_n\}$ be a path that enables a user P to achieve an activity. P 's opinion on the nodes $\{N_i\}_{i \in [1..n]}$ for an activity are denoted by $O_{N_i} = (b_{N_i}, d_{N_i}, u_{N_i}, a_{N_i})$. P 's opinion on the path σ for achieving an activity, denoted by $O_\sigma = (b_\sigma, d_\sigma, u_\sigma, a_\sigma)$, can be derived by the conjunction of P 's opinions on $\{N_i\}_{i \in [1..n]}$. $O_{\sigma = \{N_1, \dots, N_n\}} = \wedge \{O_{N_i}\}_{i \in [1..n]}$. Given Relations 6.1, we obtain the following generalization:

$$O_{\sigma = \{N_1, \dots, N_n\}} = \begin{cases} b_{\sigma = \{N_1, \dots, N_n\}} = b_{\wedge \{N_i\}_{i \in [1..n]}} = \prod_{i=1}^n b_{N_i} \\ d_{\sigma = \{N_1, \dots, N_n\}} = d_{\wedge \{N_i\}_{i \in [1..n]}} = 1 - \prod_{i=1}^n (1 - d_{N_i}) \\ u_{\sigma = \{N_1, \dots, N_n\}} = u_{\wedge \{N_i\}_{i \in [1..n]}} = \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) \\ a_{\sigma = \{N_1, \dots, N_n\}} = a_{\wedge \{N_i\}_{i \in [1..n]}} = \frac{\prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i})}{\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})} \end{cases} \quad (6.7)$$

The proofs of Relation 6.7 are presented in Appendix A.1.1 and the verifications of the correction (*i.e.*, $b_\sigma + d_\sigma + u_\sigma = 1$, $0 < b_\sigma < 1$, $0 < d_\sigma < 1$, $0 < u_\sigma < 1$ and $0 < a_\sigma < 1$) are in Appendix A.1.1.1.

6.2.3 Opinion on a system for an activity

After building an opinion on a path for an activity, an opinion on a system, which contains several paths to achieve an activity, can be built.

In trust propagation, to build an opinion on a target node in a graph, the consensus operator is used because it represents the consensus of the opinions of different persons through different paths [JHPo6, JBo8]. Whereas, in our work, an opinion on a system is the opinion of a person on one or several paths. If $O_x^P = (b_x^P, d_x^P, u_x^P, a_x^P)$ is P 's opinion on x and $O_y^P = (b_y^P, d_y^P, u_y^P, a_y^P)$ is P 's opinion on y , $O_{x \vee y}^P$ represents P 's opinion on x or y or both. Thus, the disjunction operator is the appropriate operator to evaluate an opinion on a system. In the following, we show how to build an opinion on a system when (i) the system has only independent paths and (ii) the system has dependent paths.

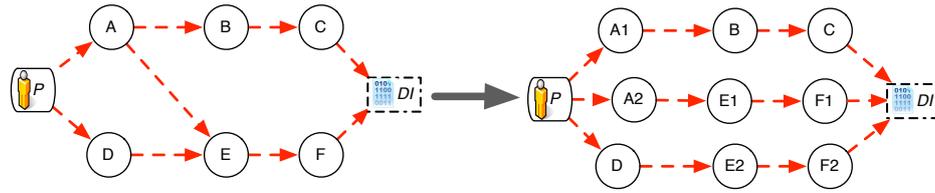


Figure 6.2 – Graph transformation using node splitting.

6.2.3.1 Independent paths

Let $\{\sigma_1, \sigma_2, \dots, \sigma_m\}$ be the paths that enable a user P to achieve an activity. The user's opinion on the paths $\{\sigma_i\}_{i \in [1..m]}$ for an activity are denoted by $O_{\sigma_i} = (b_{\sigma_i}, d_{\sigma_i}, u_{\sigma_i}, a_{\sigma_i})$.

The user opinion on the system α for achieving the activity, denoted by $O_\alpha = (b_\alpha, d_\alpha, u_\alpha, a_\alpha)$ can be derived by the disjunction of P 's opinions in $\{\sigma_i\}_{i \in [1..m]}$. Thus, $O_\alpha = \bigvee \{O_{\sigma_i}\}_{i \in [1..m]}$. Given Relation 6.3, we obtain the following generalization:

$$O_{\alpha=\{\sigma_1, \dots, \sigma_m\}} = \begin{cases} b_{\alpha=\{\sigma_1, \dots, \sigma_m\}} = b_{\bigvee \{\sigma_i\}} = 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) \\ d_{\alpha=\{\sigma_1, \dots, \sigma_m\}} = d_{\bigvee \{\sigma_i\}} = \prod_{i=1}^m d_{\sigma_i} \\ u_{\alpha=\{\sigma_1, \dots, \sigma_m\}} = u_{\bigvee \{\sigma_i\}} = \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) \\ a_{\alpha=\{\sigma_1, \dots, \sigma_m\}} = a_{\bigvee \{\sigma_i\}} = \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})} \end{cases} \quad (6.8)$$

The proofs of Relations 6.8 are presented in Appendix A.1.2.

6.2.3.2 Dependent paths

In subjective logic as in probabilistic logic, the disjunction is not distributive over the conjunction, *i.e.*, we have $O_x \wedge (O_y \vee O_z) \neq (O_x \wedge O_y) \vee (O_x \wedge O_z)$. This is due to the fact that opinions must be assumed to be independent, whereas distribution always introduces an element of dependence. In SOCIOTRUST, this problem has been resolved by using conditional probability. There is not yet an equivalent to the conditional probability, which is applied in the standard probabilistic logic, in subjective logic.

In order to apply subjective logic for evaluating trust in a system, we propose to transform a graph having dependent paths to a graph having independent paths. Once this transformation is made, we can apply the Relations 6.7 and 6.8. To do that, two methods are proposed **Copy** and **Split**.

Copy: this method is achieved by duplicating a common node into several different nodes as illustrated in Figure 6.2. The left side of this figure shows an example of a graph that has three dependent paths. The source node is P and the target node is DI . The dependent paths are: $\sigma_1 = \{A, B, C\}$, $\sigma_2 = \{A, E, F\}$ and $\sigma_3 = \{D, E, F\}$. The common nodes are A , E and F . For instance, A is a common node between σ_1 and σ_2 . By applying **Copy**, A becomes A_1, A_2 such that in the new graph, $A_1 \in \sigma'_1 = \{A_1, B, C\}$ and $A_2 \in \sigma'_2 = \{A_2, E, F\}$, so is the case for the nodes E and F . The right part of Figure 6.2 shows the new graph after duplicating the common nodes. The new graph contains the paths $\sigma'_1 = \{A_1, B, C\}$, $\sigma'_2 = \{A_2, E_1, F_1\}$ and $\sigma'_3 = \{D, E_2, F_2\}$. Concerning

Algorithm 1: Copy algorithm.

Find all the paths $\sigma_{i:i \in [1..n]}$ for an activity performed by a person

```

foreach  $\sigma_{i:i \in [1..n]}$  do
  foreach  $N_{j:j \in [1..length(\sigma_i)]} \in \sigma_i$  do
    foreach  $k \neq i: N_j \in \sigma_k$  do
      Create a node  $N_{ik}$ 
       $O_{N_{ik}} \leftarrow O_{N_j}$ 
      Replace  $N_j$  by  $N_{ik}$  in  $\sigma_k$ 

```

opinions, we keep the same opinion associated to the original node on the duplicated nodes. This method is based on the idea that the new produced path σ' maintains the same opinion of the original path σ . In this case $O_{\sigma_1} = O_{\sigma'_1}$ and $O_{\sigma_2} = O_{\sigma'_2}$. This method is shown in Algorithm 1.

Split: similar to **Copy**, nodes are duplicated to obtain independent paths as shown in Figure 6.2. In order to maintain the opinion on the global system, we split the opinion on the dependent node into independent opinions, such that their disjunction produces the original opinion. Formally speaking, if node A is in common between σ_1 and σ_2 and the opinion on A is O_A , A is duplicated into $A_1 \in \sigma'_1$ and $A_2 \in \sigma'_2$ and the opinion O_A is split into O_{A_1} and O_{A_2} where O_{A_1} and O_{A_2} satisfy the following relations: $O_{A_1} = O_{A_2}$ and $O_{A_1} \vee O_{A_2} = O_A$. The following are the obtained split opinion in two cases, the case of splitting an opinion into two independent opinions and the case of splitting an opinion into n independent opinions.

- Splitting a dependent opinion into two independent opinions:

$$\begin{aligned}
 & \bigwedge \left\{ \begin{array}{l} O_{A_1} \vee O_{A_2} = O_A \\ O_{A_1} = O_{A_2} \end{array} \right. \Leftrightarrow \\
 & \left\{ \begin{array}{l} b_{A_1} \vee b_{A_2} = b_A \\ d_{A_1} \vee d_{A_2} = d_A \\ u_{A_1} \vee u_{A_2} = u_A \\ a_{A_1} \vee a_{A_2} = a_A \end{array} \right. \bigwedge \left\{ \begin{array}{l} b_{A_1} = b_{A_2} \\ d_{A_1} = d_{A_2} \\ u_{A_1} = u_{A_2} \\ a_{A_1} = a_{A_2} \end{array} \right. \Rightarrow \\
 & \left\{ \begin{array}{l} b_{A_1} = b_{A_2} = 1 - \sqrt{1 - b_A} \\ d_{A_1} = d_{A_2} = \sqrt{d_A} \\ u_{A_1} = u_{A_2} = \frac{\sqrt{d_A} + u_A - \sqrt{d_A}}{2} \\ a_{A_1} = a_{A_2} = \frac{\sqrt{1 - b_A} - \sqrt{1 - b_A - a_A u_A}}{\sqrt{d_A + u_A} - \sqrt{d_A}} \end{array} \right. \quad (6.9)
 \end{aligned}$$

- Splitting a dependent opinion into n independent opinions:

$$\begin{aligned}
 & \bigwedge \left\{ \begin{array}{l} O_{A_1} \vee O_{A_2} \vee \dots \vee O_{A_n} = O_A \\ O_{A_1} = O_{A_2} = \dots = O_{A_n} \end{array} \right. \Leftrightarrow \\
 & \left\{ \begin{array}{l} b_{A_1} \vee b_{A_2} \vee \dots \vee b_{A_n} = b_A \\ d_{A_1} \vee d_{A_2} \vee \dots \vee d_{A_n} = d_A \\ u_{A_1} \vee u_{A_2} \vee \dots \vee u_{A_n} = u_A \\ a_{A_1} \vee a_{A_2} \vee \dots \vee a_{A_n} = a_A \end{array} \right. \bigwedge \left\{ \begin{array}{l} b_{A_1} = b_{A_2} = \dots = b_{A_n} \\ d_{A_1} = d_{A_2} = \dots = d_{A_n} \\ u_{A_1} = u_{A_2} = \dots = u_{A_n} \\ a_{A_1} = a_{A_2} = \dots = a_{A_n} \end{array} \right. \Rightarrow
 \end{aligned}$$

Algorithm 2: Split algorithm.

Find all the paths $\sigma_{i:i \in [1..n]}$ for an activity performed by a person

```

foreach  $\sigma_{i:i \in [1..n]}$  do
  foreach  $N_{j:j \in [1..length(\sigma_i)]} \in \sigma_i$  do
    foreach  $k \neq i: N_j \in \sigma_k$  do
      Create a node  $N_{ik}$ 
       $O_{N_{ik}} \leftarrow$  opinion resulted from Relation 6.10
      Replace  $N_j$  by  $N_{ik}$  in  $\sigma_k$ 

```

$$\begin{cases} b_{A1} = b_{A2} = \dots = b_{A_n} = 1 - (1 - b_A)^{\frac{1}{n}} \\ d_{A1} = d_{A2} = \dots = d_{A_n} = d_A^{\frac{1}{n}} \\ u_{A1} = u_{A2} = \dots = u_{A_n} = (d_A + u_A)^{\frac{1}{n}} - d_A^{\frac{1}{n}} \\ a_{A1} = a_{A2} = \dots = a_{A_n} = \frac{(1-b_A)^{\frac{1}{n}} - (1-b_A - a_A u_A)^{\frac{1}{n}}}{(d_A + u_A)^{\frac{1}{n}} - d_A^{\frac{1}{n}}} \end{cases} \quad (6.10)$$

The proofs of Relations 6.9 and 6.10 are provided in A.1.3. **Split** is formalized in Algorithm 2.

Next section presents the experimental evaluation.

6.3 EXPERIMENTAL EVALUATION

In this section, we compare **Copy** and **Split** to a modified version of TNA-SL [JHP06], that is based on simplifying the graph by deleting the dependent paths that have high value of uncertainty (*cf.* Section 3.3.2). In TNA-SL, after the graph simplification, trust is propagated. In our work, trust is not propagated and a comparison to a propagation approach has no sense. Thus, we modify TNA-SL such that trust evaluation is made by applying Relations 6.7 and 6.8 introduced in Section 6.2. We call this method a modified TNA-SL (**mTNA**).

We present different experiments, their results, analysis and interpretation. The main objectives for conducting the experiments are: (i) to compare the proposed methods and evaluating their accuracy and (ii) to confront this approach with real users. The first two experiments are related to the first objective while the third experiment is devoted to the second objective. Next sections present the different experiments, their results and analysis.

6.3.1 Comparing the proposed methods

To tackle the first objective, we experiment with a graph that contains only independent paths. The three methods, **mTNA**, **Copy** and **Split** give the same exact results as expected because the three of them follow the same computational model when graphs contain only independent paths. Then, we experiment on a graph that has relatively high rate of common nodes and dependent paths. 75% of the paths of the chosen graph are dependent paths and 60% of nodes are common nodes.

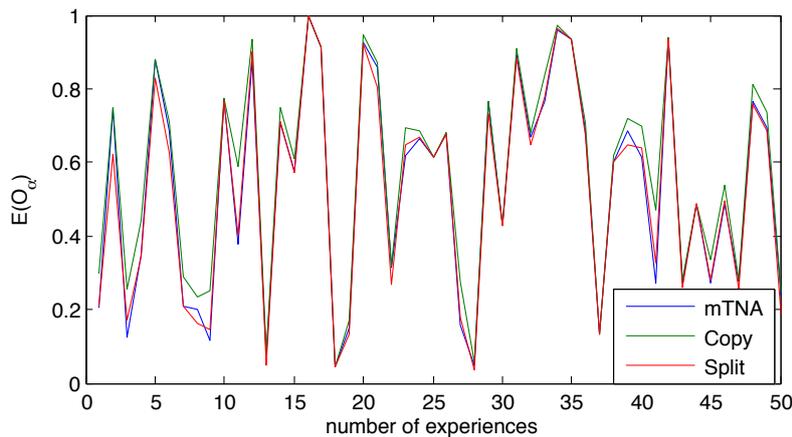


Figure 6.3 – Value of the probability expectation for 50 persons using the three methods **mTNA**, **Copy** and **Split**.

In our experiments, random opinions $O_N = (b_N, d_N, u_N, a_N)$ are associated to each node, and the opinion's probability expectation value of the graph, $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha$ is computed using the three methods, **mTNA**, **Copy** and **Split**. This experiment is repeated 50 times where each time represents random opinions of a person associated to the different nodes that compose the graph. We analyze the opinion's probability expectation values of the graph, $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha$ and not all the opinion parameters $O_\alpha = (b_\alpha, d_\alpha, u_\alpha, a_\alpha)$ for simplicity.

Figure 6.3 shows obtained results. We notice that the three methods almost have the same behavior, when the $\mathbb{E}(O_\alpha)$ increases in one method it increases in the other methods, and vice versa. We also observe some differences among the three methods that are not always negligible like at experience 9 and 40 in Figure 6.3. This observation led us to the question: which of these methods give the most accurate results? To evaluate the accuracy of **Split**, **Copy** and **mTNA**, we conduct other experiments explained in the next section.

6.3.2 Studying the accuracy of the proposed methods

SocioTrust that uses theory of probability to evaluate trust in a system, has the advantages that it has no approximations in case there are dependent paths thanks to conditional probability (*cf.* Chapter 5). Thus it works perfectly if users are sure of their judgments of trust *i.e.*, the values of uncertainty are equal to 0.

Subjective logic is equivalent to traditional probabilistic logic when $b + d = 1$ such that $u = 0$, *i.e.*, the value of uncertainty is equal to 0. When $u = 0$, the operations in subjective logic are directly compatible with the operations of the traditional probability. In this case the value of $\mathbb{E}(O) = b + au = b$ corresponds to the value of probability.

Since SocioTrust is based on probability theory, the obtained results by applying subjective logic if $u = 0$ should be equal to the ones using probability theory. We can evaluate the accuracy of the proposed methods by setting $u = 0$ and comparing the value of $b_\alpha = \mathbb{E}(O_\alpha)$ resulted

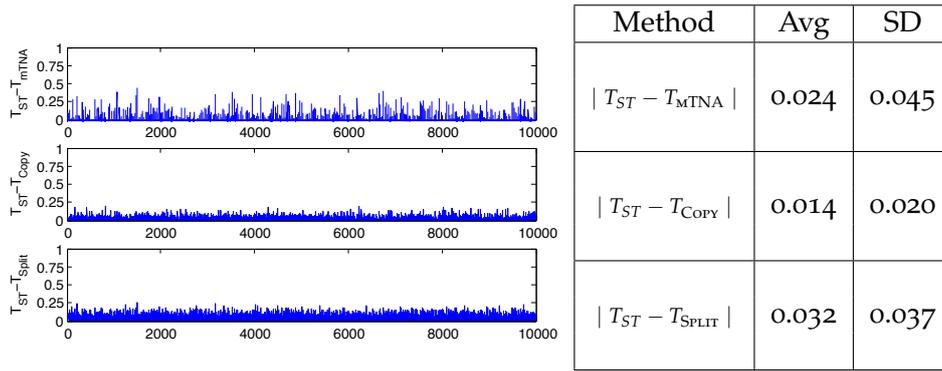


Figure 6.4 – The difference between the opinion’s probability expectation of a graph using **mTNA**, **Copy** and **Split** when $u = 0$ and the trust value resulting from using **SOCIOTRUST**.

from applying the three methods to the trust value obtained by applying **SOCIOTRUST**.

The experiments are conducted on the graph of Section 6.3.1. Random opinions $O_N = (b_N, d_N, 0, a_N)$ are associated to each node, and the probability expectation of the graph $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha = b_\alpha$ is computed.

For simplicity, the notations T_{ST} , T_{MTNA} , T_{COPY} , T_{SPLIT} respectively denote system’s trust value resulting from applying **SOCIOTRUST** and system’s opinion probability expectation resulting from applying **mTNA**, **Copy** and **Split**.

To make our comparison of T_{ST} versus T_{MTNA} , T_{COPY} , T_{SPLIT} , we simply compute the subtractions between them *i.e.*, $T_{ST} - T_{\text{MTNA}}$, $T_{ST} - T_{\text{COPY}}$, $T_{ST} - T_{\text{SPLIT}}$. The average of each of the previous values are computed through 10000 times to obtain a reliable value. The standard deviation (*SD*) is also computed to show how much variation from the average exists in the three cases. Figure 6.4 shows obtained results.

As we notice from Figure 6.4, **Copy** is the method that gives the closest results to **SOCIOTRUST**, the average of the difference of its result when $u = 0$ and the result of traditional probability over 10000 times is equal to 0.014, which is an indication that this method gives the nearest result to the exact result and its average error rate is around 1.4%.

The average error rate of **mTNA** (2.4%) is less than **Split** (3.2%), but the standard deviation of **mTNA** is 0.045 where in **Split**, it is 0.037. That means that in some cases, **mTNA** can give results that are farther than **Split** from the exact results. Thus, **Split** shows a more stable behavior than **mTNA**.

Copy shows the most convincing result. The average error rate is around 0.014 and the standard deviation is 0.02.

The objective of this experiment is not criticizing the proposed methods in the literature for the problem of dependent paths. These methods are proposed to deal with the problem of trust propagation through a graph, whereas, in our work we focus on evaluating trust towards the whole graph. The employed operators in our case are different from the employed operators in trust propagation. TNA-SL or any proposed method in the literature can work properly in their context.

In this experiment, we show that **Copy**, our new proposed method, is the method the more adaptable to be used with respect to the context of

our work. Extensive simulations on different types of graphs are provided in Appendix A.2 and follow the same behavior presented above.

6.3.3 Social evaluation: a real case

In order to study our approach on a real case study, we need the following data: a real system modeled by using `SOCIOPATH` and real opinions held by real users. The same subpart of the `LINA` research laboratory system modeled in Section 5.4.3 is used where Figure 5.6 presents the DAG for the activity “a user accesses a document `toto` that is stored on the `SVN` server of `LINA`”, with renamed nodes A, B, C, D, E, F, G .

Since subjective logic is not used yet in real applications, users are not used to build an opinion directly using this logic. We build these opinions ourselves from users’ positive or negative observations as it is proposed in [Jøso1]. To do that, the survey introduced in Appendix A.3 is executed to collect the observations of `LINA` users about the nodes. The proposed questions collect information about the user’s usage of a node and the quantity of using it and their observations. A local opinion on each entity is built for each user (few examples are shown in Appendix A.3 as well). The opinion and the opinion’s probability expectation of the system are then computed using `Copy` for each user. The results are shown in Table 6.1.

We asked each user for a feedback about their opinion on the nodes and in the system. We were glad to notice that `LINA` users were satisfied of the obtained results, whereas in `SOCIOTRUST` approach (*cf.* Section 5.4.3), 25% of users were not satisfied of the results. In the latter approach, when users do not have enough knowledge about a node, they vote with the value 0.5, which are considered for them as neutral value. That led to incorrect data that gave a low trust value in a system. In `SUBJECTIVETRUST`, uncertainties are expressed for the opinions on the nodes and computing an opinion on a system is made considering these uncertainties. That shows that, in uncertain environments, it is more suitable to use subjective logic than probabilistic metrics for trust evaluations.

	O_A (b, d, u, a)	O_B (b, d, u, a)	O_C (b, d, u, a)	O_D (b, d, u, a)	O_E (b, d, u, a)	O_F (b, d, u, a)	O_G (b, d, u, a)	O_k (b, d, u, a)	$\mathbb{E}(O_k)$
P_1	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.99,0.01,0,0.5)	(0.83,0,0.17,0.5)	(0.99,0.01,0,0.5)	(0.83,0,0.17,0.5)	(0.6820,0,0.3810,0.8389)	0.9488
P_2	(1,0,0,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(0.83,0,0.17,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(1,0,0,)	1
P_3	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(0.99,0.01,0,0.5)	(0.6,0,0.4,0.5)	(0,0,1,0.7753)	0.7753
P_4	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(0.83,0,0.17,0.5)	(0.99,0.01,0,0.5)	(0.96,0,0.04,0.5)	(0.7888,0,0.2112,0.8604)	0.9705
P_5	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.5,0,0.5,0.5)	(0,0,1,0.7500)	0.75
P_6	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.9,0.1,0,0.5)	(0.5,0,0.5,0.5)	(1,0,0,0.5)	(0.6,0,0.4,0.5)	(0.2970,0,0.7030,0.7755)	0.8422
P_7	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.9,0.1,0,0.5)	(0.83,0,0.17,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(0.8217,0,0.1783,0.8522)	0.9736
P_8	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.9,0.1,0,0.5)	(0.5,0,0.5,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(0.5000,0,0.5000,0.8625)	0.9313
P_9	(1,0,0,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(0.95,0.05,0,0.5)	(0,0,1,0.5)	(0.99,0.01,0,0.5)	(0.96,0,0.04,0.5)	(0.9956,0,0.0044,0.7583)	0.9989
P_{10}	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.9,0.1,0,0.5)	(0,0,1,0.5)	(0.8,0.2,0,0.5)	(0.98,0,0.02,0.5)	(0,0.0047,0.9953,0.7972)	0.7984
P_{11}	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.95,0.05,0,0.5)	(0.72,0,0.28,0.5)	(0.99,0.01,0,0.5)	(0.96,0,0.04,0.5)	(0.6774,0.0001,0.3225,0.8489)	0.9512
P_{12}	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.95,0.05,0,0.5)	(0.83,0,0.17,0.5)	(0.95,0.05,0,0.5)	(1,0,0,0.5)	(0.7885,0,0.0001,0.2114,0.8301)	0.9640
P_{13}	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.95,0.05,0,0.5)	(0.83,0,0.17,0.5)	(0.95,0.05,0,0.5)	(0.83,0,0.17,0.5)	(0.6545,0.0001,0.3545,0.8110)	0.9346
P_{14}	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.99,0.01,0,0.5)	(0.72,0,0.28,0.5)	(0.99,0.01,0,0.5)	(0.72,0,0.28,0.5)	(0.5132,0,0.4868,0.8186)	0.9117
P_{15}	(1,0,0,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(0.99,0.01,0,0.5)	(0.72,0,0.28,0.5)	(0.99,0.01,0,0.5)	(0.83,0,0.17,0.5)	(0.9870,0,0.0130,0.8492)	0.9980
P_{16}	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.9,0.1,0,0.5)	(0.5,0,0.5,0.5)	(1,0,0,0.5)	(0.72,0,0.28,0.5)	(0.3564,0,0.6436,0.8011)	0.8719
P_{17}	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.99,0.01,0,0.5)	(0.83,0,0.17,0.5)	(1,0,0,0.5)	(0.83,0,0.17,0.5)	(0.6889,0,0.3111,0.8447)	0.9517
P_{18}	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.99,0.01,0,0.5)	(0.83,0,0.17,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(0.8300,0,0.1700,0.8737)	0.9785
P_{19}	(1,0,0,0.5)	(1,0,0,0.5)	(1,0,0,0.5)	(0.99,0.01,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.72,0,0.28,0.5)	(0.9196,0,0.0804,0.8525)	0.9811
P_{20}	(1,0,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.95,0.05,0,0.5)	(0,0,1,0.5)	(1,0,0,0.5)	(0.83,0,0.17,0.5)	(0,0,1,0.8836)	0.8336

Table 6.1 – Users' opinions in the system for the activity "accessing a document on the server SVN in LINA".

6.4 CONCLUSION

This chapter presented SUBJECTIVETRUST, a graph-based trust approach to evaluate user's trust in a system for an activity. We used the SOCIOPATH model to restrict a system to a DAG. We proposed an approach based on subjective logic to take users' uncertainties into account. We proposed two methods for the problem of dependent paths in graph-based trust approaches, all by using subjective logic. The necessary algorithms and relations were provided and proved. Some experiments were conducted to compare the proposed methods and evaluate their accuracy, our proposed methods showed high accuracy. A real case study was made to confront this approach to real users where all users were satisfied of the obtained results of this approach.

However, in this work, a node in the DAG represents a digital resource with the physical resources that support it and the persons who control it. A user associates each node with an opinion. It could be more interesting that a user associates each entity in a node with an opinion. In this case, three levels of trust can be evaluated, *the social trust*, *the digital trust* and *the physical trust* which are respectively the trust towards the set of persons, digital and physical resources involved in a user activity achieved through a system.

CONCLUSION AND PERSPECTIVES

7

“What we call the beginning is often the end. And to make an end is to make a beginning. The end is where we start from.”

-T.S. Eliot.

Digital activities are achieved everyday by users through different systems. When users need to choose a system for a particular activity, they evaluate it considering many criteria like QoS, economical aspects, *etc.* This thesis focused on enlightening some aspects about the used system to help the user to choose a system that satisfies her expectations. The aspects we focused on are the user’s digital and social dependences in a system for an activity, their degrees and the level of a user’s trust towards the used system. To realize this approach, we fixed two main objectives:

1. Proposing a model that formalizes a system considering the different entities that compose it (physical, digital or social entities) and the relations between them.
2. Evaluating trust in a system for an activity based on this model.

By focusing on the proposed models that formalize a system in the literature, we found out that these models are complex and mostly done to retrieve information about the used digital or physical entities in a system, and ignore the social entities like the involved persons in a system.

In this thesis, we proposed SOCIOPATH, a simple model that allows to formalize the entities in a system and the relations between them. In this contribution, we observed that the entities that compose a digital system can be digital, physical or human entities. We defined a model that formalizes all these entities and the relations between them. We provided this model with the rules that discover some implicit relations in a system and enriched it with definitions that illustrate some main concepts about the used system. SOCIOPATH allows to answer the user of some main questions that concern her used system.

Trust works in the literature focus on one granularity of trust; trusting a person, a product, a resource, *etc.* That reflects one entity in a used system. Trusting a system as a composition of a set of entities and relations between them has not been studied deeply.

By focusing on trust works existing in the literature, one direction drew our attention. This direction is trust propagation in social networks. This

approach aims to propagate trust between two nodes in a graph that represents a social network. The propagated trust value results from combining trust values through this graph.

From SOCIOPATH models, we can obtain a directed acyclic graph (DAG) where nodes represent a set of entities that plays a role for achieving the users' activity and the set of edges represents the paths a user follows to achieve her activity.

Based on this DAG and inspired from graph-based trust approaches proposed for the problem of trust propagation, we proposed two approaches, SOCIOTRUST and SUBJECTIVETRUST, to evaluate trust in a system for an activity. SOCIOTRUST is based on probability theory which can be applied in full-knowledge environments where users do not need to express their uncertainty, and SUBJECTIVETRUST is based on subjective logic which can be applied in uncertain environments, this approach allows users to express their uncertainties. The necessary relations and algorithms for combining the trust values towards the entities in the DAG have been provided and proved, and experiments have been conducted to validate these approaches.

PERSPECTIVES

This work opens several perspectives that can be related to the model SOCIOPATH or related to the trust evaluation.

SocioPath perspectives: SOCIOPATH focuses on the user's needs without taking into account the other requirement in a system. For instance, the SOCIOPATH modeling requires revealing some important information about a system, which goes against the privacy requirement of the other participants in a system. This can be seen as a privacy breach. Persons in a system should be aware of this privacy concern, and work should be done to meet their privacy requirements.

SOCIOPATH is not only restricted to evaluate trust, it can be developed to be used in different applications and for different objectives. It can be used to point out accesses and controls relations within an architecture. This is particularly useful to check whether the system respects the trust and the privacy expected by its users. Being able to test an architecture compliance with respect to users' privacy policies and trust requirements can be a target to follow in a future work.

A further line of research may be devoted to investigate the amount of information about the system, that is needed to derive hidden relations by applying SOCIOPATH. The service level agreements of the system's components, rather than inner design and implementation details (that may not be available or disclosed), should be enough to draw meaningful conclusions.

Besides that SOCIOPATH allows transforming an activity achieved through a system to a DAG, and it allows to deduce the relations of digital and social dependences and their degrees on the entities of a system architecture for an activity as we show in Chapter 4. Currently SOCIOPATH does not distinguish the different kinds of access and control of an actor

towards a digital resource. In order to consider intentions and expectations of users regarding digital resources, SOCIOPATH can be enriched with access and control typologies, to define different kinds of dependences. Moreover, no difference is made between what persons can do and what they are allowed to do according to the law, the moral rules, *etc.* We aim at distinguishing between dependences related to the system's architecture, and dependences related to social commitments.

Trust perspectives: The contribution of evaluating trust can be also developed. Transforming the SOCIOPATH models into DAGs required some simplification in node representations, where a node represents a digital resource with the physical resources that support it and the persons who control it. A user associates each node with a single value of trust that represents her trust towards this node as a combination of these entities.

This simplification might be unnecessary and a user can associate each entity in the system (physical, digital, or person) with a trust level. In this case, three levels of trust can be studied, (i) *a social trust in a system for an activity* that is computed from combining the trust values associated to the persons, (ii) *a digital trust in a system for an activity* that is computed from combining the trust values associated to the digital resources and (iii) *a physical trust in a system for an activity* that is computed from combining the trust values associated to the physical resources.

These three levels of trust require extensions in the SOCIOPATH meta-model, where three relations of trust can be added from a person to respectively a physical, a digital resource and a person and require also changes in the obtained DAG. In the actual version, a node in the DAG is an artifact with the physical resources that support it and the persons who control it. In the extension we aim to make, we can obtain three different DAGs. The nodes in the first DAG are only the digital resources that allow a person to achieve an activity. This DAG will allow to compute the *digital trust in a system for an activity*. The nodes in the second DAG are only the physical resources that allow a person to achieve an activity. This DAG will allow to compute the *physical trust in a system for an activity*. The nodes in the third DAG are only the persons that allow a user to achieve an activity. This DAG will allow to compute the *social trust in a system for an activity*.

The provided relations in SOCIOTRUST and SUBJECTIVETRUST can be applied on these three DAGs to obtain these three levels of trust.

LIST OF FIGURES

2.1	Enterprise architecture layers.	10
2.2	Interactions between TOGAF metamodel, stakeholders and the models.	11
2.3	A simplified version of TOGAF metamodel.	12
2.4	System/Organization Matrix [Tog09a].	13
2.5	Application&User Location diagram.	14
2.6	OBASHI B&IT diagram [OBA].	16
2.7	OBASHI dataflow.	17
3.1	Mcknight's trust model [MCC98].	22
3.2	Mayer's trust model [RCMS95].	23
3.3	Opinion space.	26
3.4	Trust relationships in social network.	27
3.5	A general method for trust propagation through a graph.	29
3.6	The problem of the graphs that has dependent paths $T_B^A \otimes ((T_C^B \otimes T_E^C) \oplus (T_D^B \otimes T_E^D)) \neq (T_B^A \otimes T_C^B \otimes T_E^C) \oplus (T_B^A \otimes T_D^B \otimes T_E^D)$	30
3.7	Proposed solutions by Jøsang <i>et al.</i> to transform a graph having dependent paths to a graph having independent paths. (a) Eliminating the uncertain paths (TNA-SL). (b) Splitting the common edges and the opinions associated to them, $T = T_1 \oplus T_2$	32
4.1	Graphical view of SOCIOPATH as a UML class diagram.	36
4.2	Use case example: a document accessed by 2 different operating systems.	38
4.3	The relation provide.	39
4.4	The relations of access/control in the example: a document accessed by 2 different operating systems.	42
4.5	GoogleDocs snapshot.	47
4.6	Relations of access and control in the architecture GoogleDocs.	48
4.7	Degree of dependence on persons' sets.	50
5.1	The node the WDAG.	54
5.2	The activity "John accesses a document toto" as a DAG.	55
5.3	The activity "John accesses a document toto" as a WDAG.	55
5.4	System trust value according to the length of paths.	61
5.5	System trust value according to the number of paths.	61
5.6	LINA's WDAG for the activity "accessing a document toto on the SVN".	62

6.1	The activity "John accesses a document <code>toto</code> " as a WDAG where the weights are opinions.	67
6.2	Graph transformation using node splitting.	71
6.3	Value of the probability expectation for 50 persons using the three methods mTNA , Copy and Split	74
6.4	The difference between the opinion's probability expectation of a graph using mTNA , Copy and Split when $u = 0$ and the trust value resulting from using SOCIOTrUST	75

List of Tables

3.1	The different obtained results of Relations 3.1, 3.2 by applying an example of discrete metrics and continuous metrics on a simple graph	31
4.1	Glossary of notations (1).	39
4.2	Sets of persons John depends on.	49
5.1	Glossary of notations (2).	56
5.2	Different systems and their trust value.	60
5.3	User's trust value in the system SVN in LINA.	62
6.1	Users' opinions in the system for the activity "accessing a document on the server SVN in LINA".	77
A.1	Graphs that have different topologies.	116
A.2	Different graphs and their values of the probability expectation for 50 persons using the three methods Copy , Split and mTNA	117
A.3	The difference between the opinion's probability expectation of a graph using SUBJECTIVETRUST and the trust value resulting of using SOCIOTRUST.	118

BIBLIOGRAPHY

- [AABR13] Manuel Atencia, Mustafa Al-Bakri, and Marie-Christine Rousset. Trust in Networks of Ontologies and Alignments. *Knowledge and Information Systems*, pages 1–27, 2013. (Cited on page 26.)
- [ABAR12] Mustafa Al-Bakri, Manuel Atencia, and Marie-Christine Rousset. TrustMe, I Got What You Mean! - A Trust-Based Semantic P2P Bookmarking System. In *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 442–445, 2012. (Cited on page 26.)
- [ABSAL13] Nagham Alhadad, Yann Busnel, Patricia Serrano-Alvarado, and Philippe Lamarre. Graph-Based Trust Model for Evaluating Trust Using Subjective Logic. Technical Report hal-00871138, LINA – CNRS : UMR6241, 2013. (Cited on pages 3 et 67.)
- [ACH96] Cristina Aurrecochea, Andrew T. Campbell, and Linda Hauw. A Survey of QoS Architectures. *Multimedia Systems*, 6(3):138–151, 1996. (Cited on page 35.)
- [AFGL08] Isaac Agudo, Carmen Fernandez-Gago, and Javier Lopez. A Model for Trust Metrics Analysis. In *Proceedings of the 5th International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*, pages 28–37, 2008. (Cited on pages 27, 28 et 31.)
- [AGo7] Donovan Artz and Yolanda Gil. A Survey of Trust in Computer Science and the Semantic Web. *Web Semantic*, 5(2):58–71, 2007. (Cited on page 26.)
- [ALB⁺11a] Nagham Alhadad, Philippe Lamarre, Yann Busnel, Patricia Serrano-Alvarado, and Marco Biazzi. SocioPath: In Whom You Trust? In *Journées Bases de Données Avancées*, Rabat, Morocco, 2011. (Cited on page 2.)
- [ALB⁺11b] Nagham Alhadad, Philippe Lamarre, Yann Busnel, Patricia Serrano-Alvarado, Marco Biazzi, and Christophe Sibertin-Blanc. SOCIOPATH: In Whom You Trust? Technical report, LINA – CNRS : UMR6241, September 2011. (Cited on page 2.)
- [ALB⁺12] Nagham Alhadad, Philippe Lamarre, Yann Busnel, Patricia Serrano-Alvarado, Marco Biazzi, and Christophe Sibertin-Blanc. SocioPath: Bridging the Gap between Digital and Social

- Worlds. In *Proceedings of the 23rd International Conference on Database and Expert Systems Applications (Dexa)*, pages 497–505, 2012. (Cited on pages 2 et 35.)
- [ARH00] Alfarez Abdul-Rahman and Stephen Hailes. Supporting Trust in Virtual Communities. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS)*, 2000. (Cited on page 23.)
- [ASABL13] Nagham Alhadad, Patricia Serrano-Alvarado, Yann Busnel, and Philippe Lamarre. Trust Evaluation of a System for an Activity. In *10th International Conference on Trust, Privacy & Security in Digital Business (TrustBus)*, pages 24–36, 2013. (Cited on pages 3 et 53.)
- [CH97] Bruce Christianson and William S. Harbison. Why Isn't Trust Transitive? In *Proceedings of the 5th International Workshop on Security Protocols*, pages 171–176, 1997. (Cited on page 23.)
- [CNS03] Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A Formal Model for Trust in Dynamic Networks. In *Proceedings of the 1st International Conference on Software Engineering and Formal Methods (SEFM)*, pages 54–61, 2003. (Cited on page 23.)
- [Coo01] Karen Cook. *Trust in Society*. New York: Russell Sage Foundation, 2001. (Cited on page 21.)
- [Dep96] Department of Defense. Technical Architecture Framework for Information Management TAFIM. <http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA321171>, April 1996. (Cited on page 10.)
- [Deu62] Morton Deutsch. Cooperation and trust: Some theoretical notes. 1962. (Cited on page 22.)
- [Fuk96] Francis Fukuyama. *Trust: The Social Virtues and the Creation of Prosperity*. Touchstone Books, 1996. (Cited on page 21.)
- [Gam00] Diego Gambetta. Can we Trust Trust. *Trust: Making and breaking cooperative relations*, pages 213–237, 2000. (Cited on pages 22, 24 et 56.)
- [GH06] Jennifer Golbeck and James A. Hendler. Inferring Binary Trust Relationships in Web-Based Social Networks. *ACM Transactions on Internet Technology*, 6(4):497–529, 2006. (Cited on pages 23, 28 et 31.)
- [Gia10] Ronald E. Giachetti. *Design of Enterprise Systems: Theory, Architecture, and Methods*. CRC Press, Boca Raton Florida, 2010. (Cited on page 9.)
- [Gol05] Jennifer Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, Department of Computer Science, University of Maryland, 2005. (Cited on pages 22, 26, 28 et 32.)

- [Golo6] Jennifer Golbeck. Trust on the World Wide Web: a Survey. *Foundations and Trends in Web Science*, 1(2):131–197, January 2006. (Cited on page 26.)
- [Har07] Rachel Harrison. *TOGAF Version 8.1*. Van Haren Publishing, New York, 2007. (Cited on pages 10 et 11.)
- [Hew06] Niles E Hewlett. The USDA Enterprise Architecture Program. PMP CEA, Enterprise Architecture Team, USDA-OCIO, 2006. (Cited on page 9.)
- [HWS09] Chung-Wei Hang, Yonghong Wang, and Munindar P. Singh. Operators for Propagating Trust and their Evaluation in Social Networks. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1025–1032, 2009. (Cited on page 26.)
- [JBo8] Audun Jøsang and Touhid Bhuiyan. Optimal Trust Network Analysis with Subjective Logic. In *Proceeding of the 2nd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, pages 179–184, 2008. (Cited on pages 26, 29, 30 et 70.)
- [JHP06] Audun Jøsang, Ross Hayward, and Simon Pope. Trust Network Analysis with Subjective Logic. In *Proceedings of the 29th Australasian Computer Science Conference (ACSC)*, pages 85–94, 2006. (Cited on pages 26, 29, 30, 31, 70 et 73.)
- [JIB07] Audun Jøsang, Roslan Ismail, and Colin Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2):618–644, 2007. (Cited on pages 21, 22, 24, 26, 53 et 55.)
- [Jøso1] Audun Jøsang. A Logic for Uncertain Probabilities. *Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, 2001. (Cited on pages 24, 25, 62, 67, 68, 76, 114 et 119.)
- [Jøso7] Audun Jøsang. Trust and Reputation Systems. In *Proceeding of the 7th International School on Foundations of Security Analysis and Design (FOSAD)*, pages 209–245, 2007. (Cited on pages 23 et 24.)
- [Jos09] Andrew Josey. *TOGAF Version 9: A Pocket Guide*. Van Haren Publishing, 2 edition, 2009. (Cited on page 11.)
- [KFJ02] Lalana Kagal, Tim Finin, and Anupam Joshi. Developing Secure Agent Systems Using Delegation Based Trust Management. In *the 2nd International Workshop of Security of Mobile Multi-Agent Systems (SEMAS) held at the International Foundation for Autonomous Agents and Multiagent Systems (AAMAS)*, 2002. (Cited on page 23.)

- [LABW92] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in Distributed Systems: Theory and Practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992. (Cited on page 35.)
- [LW10] Lei Li and Yan Wang. Subjective Trust Inference in Composite Services. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, 2010. (Cited on pages 24, 56 et 68.)
- [LW11a] Lei Li and Yan Wang. A Subjective Probability Based Deductive Approach to Global Trust Evaluation in Composite Services. In *Proceedings of the 9th IEEE International Conference on Web Services (ICWS)*, pages 604–611, 2011. (Cited on page 26.)
- [LW11b] Lei Li and Yan Wang. A subjective probability based deductive approach to global trust evaluation in composite services. In *Proceedings of the 9th IEEE International Conference on Web Services (ICWS)*, pages 604–611, Washington, DC, USA, 2011. IEEE Computer Society. (Cited on page 68.)
- [LWOL11] G. Liu, Y. Wang, M. Orgun, and E. Lim. Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks. *Services Computing, IEEE Transactions on*, PP(99):1, 2011. (Cited on page 31.)
- [MA05] Paolo Massa and Paolo Avesani. Controversial Users Demand Local Trust Metrics: an Experimental Study on Epinions.com Community. In *Proceedings of the 20th National Conference on Artificial intelligence (AAAI)*, pages 121–126, 2005. (Cited on pages 26 et 32.)
- [Man98] Daniel W. Manchala. Trust Metrics, Models and Protocols for Electronic Commerce Transactions. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS)*, pages 312–321, 1998. (Cited on page 23.)
- [Mar94] Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994. (Cited on pages 21, 26 et 53.)
- [MC96] D. Harrison Mcknight and Norman L. Chervany. The Meanings of Trust. Technical report, University of Minnesota, Carlson School of Management, 1996. (Cited on page 56.)
- [MCC98] D. Harrison Mcknight, Larry L. Cummings, and Norman L. Chervany. Initial Trust Formation in New Organizational Relationships. *Academy of Management Review*, 23(3):473–490, 1998. (Cited on pages 21, 22 et 82.)
- [McLo6] Carolyn McLeod. *Trust*. The Stanford Encyclopedia of Philosophy, 2006. (Cited on page 21.)

- [MFGL12] Francisco Moyano, M. Carmen Fernández-Gago, and Javier Lopez. A Conceptual Framework for Trust Models. In *Proceedings of the 9th International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*, pages 93–104, 2012. (Cited on pages 21 et 53.)
- [MGM06] Sergio Marti and Hector Garcia-Molina. Taxonomy of Trust: Categorizing P2P Reputation Systems. *Computer Network Journal*, 50(4):472–484, 2006. (Cited on page 35.)
- [MSF98] Michael Gruninger Mark S. Fox. Enterprise Modeling. *AI Magazine*, 19(3):109–121, 1998. (Cited on page 10.)
- [OBA] The official OBASHI Website. <http://www.obashi.co.uk/>. (Cited on pages 10, 13, 14, 16 et 82.)
- [OBA10] OBASHI. OBASHI Explained. <http://www.apmg-international.tv/qualifications/specialist-qualifications/obashi/item/obashi-explained>, 2010. (Cited on page 16.)
- [RAD03] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust Management for the Semantic Web. In *Proceedings of the 2nd International Semantic Web Conference (ISWC)*, pages 351–368, 2003. (Cited on pages 26, 27 et 31.)
- [RCMS95] James H. Davis Roger C. Mayer and F. David Schoorman. An Integrative Model of Organizational Trust. *The Academy of Management Review*, 20(3):709–734, 1995. (Cited on pages 22, 23 et 82.)
- [RHZ85] John K. Rempel, John G. Holmes, and Mark P. Zanna. Trust in close relationships. *Personality and Social Psychology*, 49(1):95–112, 1985. (Cited on page 21.)
- [Roh05] Michael Rohloff. Enterprise Architecture-Framework and Methodology for the Design of Architectures in the Large. In *Proceedings of the 13th European Conference on Information Systems (ECIS)*, pages 1659–1672, 2005. (Cited on page 10.)
- [SKJ⁺00] Jay Schneider, Gerd Kortuem, Joe Jager, Steve Fickas, and Zary Segall. Disseminating Trust Information in Wearable Communities. *Personal Ubiquitous Computing*, 4(4):245–248, 2000. (Cited on page 23.)
- [SR07] Carolyn Strano and Qamar Rehmani. The Role of the Enterprise Architect. *Information Systems and e-Business Management*, 5(4):379–396, 2007. (Cited on page 9.)
- [TA12] Mozghan Tavakolifard and Kevin C. Almeroth. A Taxonomy to Express Open Challenges in Trust and Reputation Systems. *Communications*, 7(7):538–551, 2012. (Cited on page 24.)
- [The99] The Chief Information Officers Council. Federal Enterprise Architecture Framework Version 1.1. <http://www.enterprise-architecture.info/Images/Documents/Federal20EA20Framework.pdf>, September 1999. (Cited on page 10.)

- [Tog09a] Welcome to TOGAF Version 9 - Architectural Artifacts. <http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap35.html>, 2009. (Cited on pages 13 et 82.)
- [Tog09b] Welcome to TOGAF Version 9 - Content Metamodel. <http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap34.html>, 2009. (Cited on page 11.)
- [TRA03] TRAK Enterprise Architecture. <http://trak.sourceforge.net/>, 2003. (Cited on page 10.)
- [Uslo2] Eric M. Uslaner. *The Moral Foundations of Trust*. Cambridge, UK: Cambridge University Press, 2002. (Cited on page 21.)
- [VC03] Elizabeth Gray et al Vinny Cahill. Using Trust for Secure Collaboration in Uncertain Environments. *IEEE Pervasive Computing*, 2(3):52–61, 2003. (Cited on page 23.)
- [Vil05] Lea Viljanen. Towards an Ontology of Trust. In *Proceedings of the 2nd International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*, 2005. (Cited on pages 21 et 53.)
- [Wes70] Alan F. Westin. *Privacy and Freedom*. Atheneum, New York, sixth edition edition, 1970. (Cited on page 35.)
- [YHo07] Zheng Yan and Silke Holtmanns. *Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions*, chapter Trust Modeling and Management: from Social Trust to Digital Trust. 2007. (Cited on pages 21 et 53.)
- [Zac87] John A. Zachman. A Framework for Information Systems Architecture. *IBM systems journal*, 26(3):276–292, 1987. (Cited on page 10.)
- [ZDB11] Ping Zhang, Arjan Durresi, and Leonard Barolli. Survey of Trust Management on Various Networks. In *Proceedings of the 5th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 219–226, 2011. (Cited on pages 21, 26 et 53.)

APPENDIXS

A

A.1 MATHEMATICAL PROOFS

In this section, we develop the formal proof of the introduced relations in Section 6.2 about opinion on path and opinion on a system respectively in Sections A.1.1, A.1.2. In Section A.1.3, we prove the relations of the method **Split**. In Section A.1.4, we prove that **Copy** is more optimist than **mTNA** and In Section A.1.5, we prove that **Copy** is more optimist than **Split**.

A.1.1 Opinion on a path for an activity (mathematical proof)

First, we prove the generalization proposed in Relation 6.7 introduced in Section 6.2.2 then we verify that our relations hold the properties of the opinion parameters in subjective logic in Section A.1.1.1.

$$O_{x \wedge y} = \begin{cases} b_{x \wedge y} = b_x b_y \\ d_{x \wedge y} = d_x + d_y - d_x d_y \\ u_{x \wedge y} = b_x u_y + u_x b_y + u_x u_y \\ a_{x \wedge y} = \frac{b_x u_y a_y + b_y u_x a_x + u_x a_x u_y a_y}{b_x u_y + u_x b_y + u_x u_y} \end{cases} \Rightarrow$$

$$O_{\sigma=\{N_1, \dots, N_n\}} = \begin{cases} b_{\sigma=\{N_1, \dots, N_n\}} = b_{\wedge\{N_i\}_{i \in [1..n]}} = \prod_{i=1}^n b_{N_i} \\ d_{\sigma=\{N_1, \dots, N_n\}} = d_{\wedge\{N_i\}_{i \in [1..n]}} = 1 - \prod_{i=1}^n (1 - d_{N_i}) \\ u_{\sigma=\{N_1, \dots, N_n\}} = u_{\wedge\{N_i\}_{i \in [1..n]}} = \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) \\ a_{\sigma=\{N_1, \dots, N_n\}} = a_{\wedge\{N_i\}_{i \in [1..n]}} = \frac{\prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i})}{\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})} \end{cases}$$

1. The mathematical proof of the relation b_{σ} :

Lemma A.1. $b_{\sigma=\{N_1, \dots, N_n\}} = b_{\wedge\{N_i\}_{i \in [1..n]}} = \prod_{i=1}^n b_{N_i}$

Proof. We prove by induction that, for all $n \in \mathbb{Z}^+$,

$$b_{\wedge\{N_i\}_{i \in [1..n]}} = \prod_{i=1}^n b_{N_i}$$

Base case. When $n = 2$:

$$b_{N_1 \wedge N_2} = b_{N_1} b_{N_2} = \prod_{i=1}^2 b_{N_i}$$

Induction step. Let $k \in \mathbb{Z}^+$ be given and suppose that Lemma A.1 is true for $n = k$. Then

$$b_{\wedge\{N_i\}_{i \in [1..k+1]}} = b_{\{\wedge\{N_i\}_{i \in [1..k]}\} \wedge N_{k+1}} = \prod_{i=1}^k b_{N_i} b_{N_{k+1}} = \prod_{i=1}^{k+1} b_{N_i}$$

Thus, Lemma A.1 holds for $n = k + 1$. By the principle of induction, Lemma A.1 is true for all $n \in \mathbb{Z}^+$. □

2. The mathematical proof of the relation d_{σ} :

Lemma A.2. $d_{\sigma=\{N_1, \dots, N_n\}} = d_{\wedge\{N_i\}_{i \in [1..n]}} = 1 - \prod_{i=1}^n (1 - d_{N_i})$

Proof. We prove by induction that, for all $n \in \mathbb{Z}^+$,

$$d_{\wedge\{N_i\}_{i \in [1..n]}} = 1 - \prod_{i=1}^n (1 - d_{N_i})$$

Base case. When $n = 2$:

$$\begin{aligned} d_{N_1 \wedge N_2} &= d_{N_2} + d_{N_1} - d_{N_1} d_{N_2} \\ &= 1 - (1 - d_{N_2} - d_{N_1} + d_{N_1} d_{N_2}) \\ &= 1 - (1 - d_{N_1})(1 - d_{N_2}) \\ &= 1 - \prod_{i=1}^2 (1 - d_{N_i}) \end{aligned}$$

Induction step. Let $k \in \mathbb{Z}^+$ be given and suppose that Lemma A.2 is true for $n = k$. Then

$$\begin{aligned} d_{\wedge\{N_i\}_{i \in [1..k+1]}} &= d_{\wedge\{N_i\}_{i \in [1..k]} \wedge N_{k+1}} \\ &= d_{\wedge\{N_i\}_{i \in [1..k]}} + d_{N_{k+1}} - d_{\wedge\{N_i\}_{i \in [1..k]}} d_{N_{k+1}} \\ &= \left[1 - \prod_{i=1}^k (1 - d_{N_i}) \right] + d_{N_{k+1}} - \left[1 - \prod_{i=1}^k (1 - d_{N_i}) \right] d_{N_{k+1}} \\ &= 1 - \prod_{i=1}^k (1 - d_{N_i}) + d_{N_{k+1}} - d_{N_{k+1}} + d_{N_{k+1}} \prod_{i=1}^k (1 - d_{N_i}) \\ &= 1 - \prod_{i=1}^k (1 - d_{N_i}) + d_{N_{k+1}} \prod_{i=1}^k (1 - d_{N_i}) \\ &= 1 - \left[\prod_{i=1}^k (1 - d_{N_i}) \right] [1 - d_{N_{k+1}}] \\ &= 1 - \prod_{i=1}^{k+1} (1 - d_{N_i}) \end{aligned}$$

Thus, Lemma A.2 holds for $n = k + 1$. By the principle of induction, Lemma A.2 is true for all $n \in \mathbb{Z}^+$. \square

3. The mathematical proof of the relation u_{σ} :

Lemma A.3. $u_{\sigma=\{N_1, \dots, N_n\}} = u_{\wedge\{N_i\}_{i \in [1..n]}} = \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})$

Proof. We prove by induction that, for all $n \in \mathbb{Z}^+$,

$$u_{\wedge\{N_i\}_{i \in [1..n]}} = \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})$$

Base case. When $n = 2$:

$$\begin{aligned} u_{N_1 \wedge N_2} &= b_{N_1} u_{N_2} + u_{N_1} b_{N_2} + u_{N_1} u_{N_2} \\ &= b_{N_1} u_{N_2} + u_{N_1} b_{N_2} + u_{N_1} u_{N_2} + b_{N_1} b_{N_2} - b_{N_1} b_{N_2} \\ &= (b_{N_1} + u_{N_1})(b_{N_2} + u_{N_2}) - b_{N_1} b_{N_2} \\ &= \prod_{i=1}^2 (b_{N_i} + u_{N_i}) - \prod_{i=1}^2 (b_{N_i}) \end{aligned}$$

Induction step. Let $k \in Z^+$ be given and suppose that Lemma A.3 is true for $n = k$. Then

$$\begin{aligned}
u_{\wedge\{N_i\}_{i \in [1, k+1]}} &= u_{\{\wedge\{N_i\}_{i \in [1, k]}\} \wedge N_{k+1}} \\
&= b_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} + u_{\wedge\{N_i\}_{i \in [1, k]}} b_{N_{k+1}} + u_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} \\
&= \left[\prod_{i=1}^k b_{N_i} \right] u_{N_{k+1}} + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] b_{N_{k+1}} \\
&\quad + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] u_{N_{k+1}} \\
&= \prod_{i=1}^k b_{N_i} u_{N_{k+1}} + \prod_{i=1}^k (b_{N_i} + u_{N_i}) b_{N_{k+1}} - \prod_{i=1}^k (b_{N_i}) b_{N_{k+1}} \\
&\quad + \prod_{i=1}^k (b_{N_i} + u_{N_i}) u_{N_{k+1}} - \prod_{i=1}^k (b_{N_i}) u_{N_{k+1}} \\
&= \left[\prod_{i=1}^k (b_{N_i} + u_{N_i}) \right] (b_{N_{k+1}} + u_{N_{k+1}}) - \prod_{i=1}^{k+1} (b_{N_i}) \\
&= \prod_{i=1}^{k+1} (b_{N_i} + u_{N_i}) - \prod_{i=1}^{k+1} (b_{N_i})
\end{aligned}$$

Thus, Lemma A.3 holds for $n = k + 1$. By the principle of induction, Lemma A.3 is true for all $n \in Z^+$. \square

4. The mathematical proof of the relation a_σ :

$$\mathbf{Lemma\ A.4.} \quad a_{\sigma = \{N_1, \dots, N_n\}} = a_{\wedge\{N_i\}_{i \in [1, n]}} = \frac{\prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i})}{\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})}$$

Proof. We prove by induction that, for all $n \in Z^+$,

$$a_{\wedge\{N_i\}_{i \in [1, n]}} = \frac{\prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i})}{\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})}$$

Base case. When $n = 2$:

$$\begin{aligned}
a_{N_1 \wedge N_2} &= \frac{b_{N_1} u_{N_2} a_{N_2} + b_{N_2} u_{N_1} a_{N_1} + u_{N_1} a_{N_1} u_{N_2} a_{N_2}}{b_{N_1} u_{N_2} + u_{N_1} b_{N_2} + u_{N_1} u_{N_2}} \\
&= \frac{(b_{N_1} u_{N_2} a_{N_2} + b_{N_2} u_{N_1} a_{N_1} + u_{N_1} a_{N_1} u_{N_2} a_{N_2} + b_{N_1} b_{N_2}) - b_{N_1} b_{N_2}}{u_{N_1 \wedge N_2}} \\
&= \frac{(b_{N_1} + u_{N_1} a_{N_1})(b_{N_2} + u_{N_2} a_{N_2}) - (b_{N_1} b_{N_2})}{u_{N_1 \wedge N_2}} \\
&= \frac{\prod_{i=1}^2 (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^2 (b_{N_i})}{\prod_{i=1}^2 (b_{N_i} + u_{N_i}) - \prod_{i=1}^2 (b_{N_i})}
\end{aligned}$$

Induction step. Let $k \in Z^+$ be given and suppose that Lemma A.4 is true for $n = k$. Then

$$\begin{aligned}
a_{\wedge\{N_i\}_{i \in [1, k+1]}} &= a_{\{\wedge\{N_i\}_{i \in [1, k]}\} \wedge N_{k+1}} = \\
&= \frac{b_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} a_{N_{k+1}} + b_{N_{k+1}} u_{\wedge\{N_i\}_{i \in [1, k]}} a_{\wedge\{N_i\}_{i \in [1, k]}} + u_{\wedge\{N_i\}_{i \in [1, k]}} a_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} a_{N_{k+1}}}{b_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} + u_{\wedge\{N_i\}_{i \in [1, k]}} b_{N_{k+1}} + u_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}}}
\end{aligned}$$

We denote the numerator with γ , and the denominator with β .

$$a_{\wedge\{N_i\}_{i \in [1, k+1]}} = \frac{\gamma}{\beta}$$

$$\begin{aligned} \gamma &= b_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} a_{N_{k+1}} + b_{N_{k+1}} u_{\wedge\{N_i\}_{i \in [1, k]}} a_{\wedge\{N_i\}_{i \in [1, k]}} \\ &\quad + u_{\wedge\{N_i\}_{i \in [1, k]}} a_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} a_{N_{k+1}} \\ \gamma &= \left[\prod_{i=1}^k b_{N_i} \right] u_{N_{k+1}} a_{N_{k+1}} \\ &\quad + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] \left[\frac{\prod_{i=1}^k (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^k (b_{N_i})}{\prod_{i=1}^k (b_{N_i} + u_{N_i}) - \prod_{i=1}^k (b_{N_i})} \right] b_{N_{k+1}} \\ &\quad + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] \left[\frac{\prod_{i=1}^k (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^k (b_{N_i})}{\prod_{i=1}^k (b_{N_i} + u_{N_i}) - \prod_{i=1}^k (b_{N_i})} \right] u_{N_{k+1}} a_{N_{k+1}} \\ \gamma &= \left[\prod_{i=1}^k b_{N_i} u_{N_{k+1}} a_{N_{k+1}} \right] + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] b_{N_{k+1}} \\ &\quad + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] u_{N_{k+1}} a_{N_{k+1}} \\ \gamma &= \left[\prod_{i=1}^k b_{N_i} u_{N_{k+1}} a_{N_{k+1}} \right] + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] [b_{N_{k+1}} + u_{N_{k+1}} a_{N_{k+1}}] \\ \gamma &= \left[\prod_{i=1}^k b_{N_i} u_{N_{k+1}} a_{N_{k+1}} \right] + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i} a_{N_i}) \right] [b_{N_{k+1}} + u_{N_{k+1}} a_{N_{k+1}}] \\ &\quad - \prod_{i=1}^k (b_{N_i}) [b_{N_{k+1}} + u_{N_{k+1}} a_{N_{k+1}}] \\ \gamma &= \left[\prod_{i=1}^k b_{N_i} u_{N_{k+1}} a_{N_{k+1}} \right] + \left[\prod_{i=1}^{k+1} (b_{N_i} + u_{N_i} a_{N_i}) \right] - \prod_{i=1}^k (b_{N_i}) b_{N_{k+1}} - \prod_{i=1}^k (b_{N_i}) u_{N_{k+1}} a_{N_{k+1}} \\ \gamma &= \left[\prod_{i=1}^{k+1} (b_{N_i} + u_{N_i} a_{N_i}) \right] - \prod_{i=1}^{k+1} (b_{N_i}) \\ \beta &= b_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} + u_{\wedge\{N_i\}_{i \in [1, k]}} b_{N_{k+1}} + u_{\wedge\{N_i\}_{i \in [1, k]}} u_{N_{k+1}} \\ \beta &= \left[\prod_{i=1}^k b_{N_i} \right] u_{N_{k+1}} + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] b_{N_{k+1}} \\ &\quad + \left[\prod_{i=1}^k (b_{N_i} + u_{N_i}) - \prod_{i=1}^k (b_{N_i}) \right] u_{N_{k+1}} \\ \beta &= \prod_{i=1}^k b_{N_i} u_{N_{k+1}} + \prod_{i=1}^k (b_{N_i} + u_{N_i}) b_{N_{k+1}} - \prod_{i=1}^k (b_{N_i}) b_{N_{k+1}} \\ &\quad + \prod_{i=1}^k (b_{N_i} + u_{N_i}) u_{N_{k+1}} - \prod_{i=1}^k (b_{N_i}) u_{N_{k+1}} \\ \beta &= \left[\prod_{i=1}^k (b_{N_i} + u_{N_i}) \right] (b_{N_{k+1}} + u_{N_{k+1}}) - \prod_{i=1}^{k+1} (b_{N_i}) \\ &= \prod_{i=1}^{k+1} (b_{N_i} + u_{N_i}) - \prod_{i=1}^{k+1} (b_{N_i}) \\ a_{\wedge\{N_i\}_{i \in [1, k+1]}} &= \frac{\gamma}{\beta} = \frac{\left[\prod_{i=1}^{k+1} (b_{N_i} + u_{N_i} a_{N_i}) \right] - \prod_{i=1}^{k+1} (b_{N_i})}{\prod_{i=1}^{k+1} (b_{N_i} + u_{N_i}) - \prod_{i=1}^{k+1} (b_{N_i})} \end{aligned}$$

Thus, Lemma A.4 holds for $n = k + 1$. By the principle of induction, Lemma A.4 is true for all $n \in \mathbb{Z}^+$. \square

A.1.1.1 Verifications

In this section we verify that the previous relations hold the properties of the opinion parameters in subjective logic (*i.e.*, $b + d + u = 1$ and $0 < b, d, u, a < 1$) so we verify the following:

$$\begin{cases} 0 < b_\sigma = \prod_{i=1}^n b_{N_i} < 1 \\ 0 < d_\sigma = 1 - \prod_{i=1}^n (1 - d_{N_i}) < 1 \\ 0 < u_\sigma = \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) < 1 \\ 0 < a_\sigma = \frac{\prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i})}{\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})} < 1 \\ b_\sigma + d_\sigma + u_\sigma = 1 \end{cases}$$

Lemma A.5. $0 < b_\sigma = \prod_{i=1}^n b_{N_i} < 1$

Proof.

$$\forall i \in [1..n] : 0 < b_{N_i} < 1$$

The multiplications of several values between 0 and 1 is between 0 and 1
 \Rightarrow

$$0 < \prod_{i=1}^n b_{N_i} < 1 \Rightarrow 0 < b_\sigma < 1$$

□

Lemma A.6. $0 < d_\sigma = 1 - \prod_{i=1}^n (1 - d_{N_i}) < 1$

Proof.

$$\forall i \in [1..n] :$$

$$0 < d_{N_i} < 1 \Rightarrow$$

$$1 > 1 - d_{N_i} > 0 \Rightarrow$$

$$1 > \prod_{i=1}^n (1 - d_{N_i}) > 0 \Rightarrow$$

$$0 < 1 - \prod_{i=1}^n (1 - d_{N_i}) < 1 \Rightarrow$$

$$0 < d_\sigma < 1$$

□

Lemma A.7. $0 < u_\sigma = \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) < 1$

Proof. Considering the left side of the relation, we have:

$$\forall i \in [1..n] : u_{N_i} > 0 \Rightarrow$$

$$b_{N_i} + u_{N_i} > b_{N_i} \Rightarrow$$

$$\prod_{i=1}^n (b_{N_i} + u_{N_i}) > \prod_{i=1}^n (b_{N_i}) \Rightarrow$$

$$\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) > 0 \Rightarrow$$

$$u_\sigma > 0$$

Considering the right side of the relation, we have:

$$\begin{aligned} \forall i \in [1..n] : b_{N_i} + u_{N_i} < 1 &\Rightarrow \\ \prod_{i=1}^n (b_{N_i} + u_{N_i}) < 1 &\Rightarrow \\ \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) < 1 &\Rightarrow \\ u_\sigma < 1 \end{aligned}$$

□

Lemma A.8. $0 < a_\sigma = \frac{\prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i})}{\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})} < 1$

Proof. Considering the left side of the relation, we have:

$$\left. \begin{aligned} \forall i \in [1..n] : \\ b_{N_i} + u_{N_i} a_{N_i} > b_{N_i} &\Rightarrow \prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i}) > 0 \\ b_{N_i} + u_{N_i} > b_{N_i} &\Rightarrow \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) > 0 \end{aligned} \right\} \Rightarrow$$

$$\frac{\prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i})}{\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})} > 0 \Rightarrow a_\sigma > 0$$

Considering the right side of the relation, we have:

$$\begin{aligned} \forall i \in [1..n] : 0 < a_{N_i} < 1 &\Rightarrow \\ u_{N_i} a_{N_i} < u_{N_i} &\Rightarrow \\ b_{N_i} + u_{N_i} a_{N_i} < b_{N_i} + u_{N_i} &\Rightarrow \\ \prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) < \prod_{i=1}^n (b_{N_i} + u_{N_i}) &\Rightarrow \\ \prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i}) < \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) &\Rightarrow \\ \frac{\prod_{i=1}^n (b_{N_i} + u_{N_i} a_{N_i}) - \prod_{i=1}^n (b_{N_i})}{\prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})} < 1 &\Rightarrow \\ a_\sigma < 1 \end{aligned}$$

□

Lemma A.9. $b_\sigma + d_\sigma + u_\sigma = 1$

Proof.

$$\left\{ \begin{aligned} b_\sigma &= \prod_{i=1}^n b_{N_i} \\ d_\sigma &= 1 - \prod_{i=1}^n (1 - d_{N_i}) \\ u_\sigma &= \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i}) \end{aligned} \right. \Rightarrow$$

$$b_\sigma + d_\sigma + u_\sigma = \prod_{i=1}^n b_{N_i} + 1 - \prod_{i=1}^n (1 - d_{N_i}) + \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})$$

$$b_\sigma + d_\sigma + u_\sigma = \prod_{i=1}^n b_{N_i} + 1 - \prod_{i=1}^n (b_{N_i} + u_{N_i}) + \prod_{i=1}^n (b_{N_i} + u_{N_i}) - \prod_{i=1}^n (b_{N_i})$$

$$b_\sigma + d_\sigma + u_\sigma = 1$$

□

A.1.2 Opinion on a system for an activity (mathematical proof)

We prove the generalization proposed in Relation 6.8 introduced in Section 6.2.3 then we verify that our relations hold the properties of the opinion parameters in subjective logic in Section A.1.2.1.

$$O_{x \vee y} = \begin{cases} b_{x \vee y} = b_x + b_y - b_x b_y \\ d_{x \vee y} = d_x d_y \\ u_{x \vee y} = d_x u_y + u_x d_y + u_x u_y \\ a_{x \vee y} = \frac{u_x a_x + u_y a_y - b_x u_y a_y - b_y u_x a_x - u_x a_x u_y a_y}{u_x + u_y - b_x u_y - b_y u_x - u_x u_y} \end{cases} \Rightarrow$$

$$O_{\alpha = \{\sigma_1, \dots, \sigma_m\}} = \begin{cases} b_{\alpha = \{\sigma_1, \dots, \sigma_m\}} = b_{\vee \{\sigma_i\}} = 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) \\ d_{\alpha = \{\sigma_1, \dots, \sigma_m\}} = d_{\vee \{\sigma_i\}} = \prod_{i=1}^m d_{\sigma_i} \\ u_{\alpha = \{\sigma_1, \dots, \sigma_m\}} = u_{\vee \{\sigma_i\}} = \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) \\ a_{\alpha = \{\sigma_1, \dots, \sigma_m\}} = a_{\vee \{\sigma_i\}} = \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})} \end{cases}$$

1. The mathematical proof of the relation b_α :

$$\textbf{Lemma A.10. } b_{\alpha = \{\sigma_1, \dots, \sigma_m\}} = b_{\vee \{\sigma_i\}_{i \in [1..m]}} = 1 - \prod_{i=1}^m (1 - b_{\sigma_i})$$

Proof. We prove by induction that, for all $m \in \mathbb{Z}^+$,

$$b_{\vee \{\sigma_i\}_{i \in [1..m]}} = 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) \quad (\text{A.1})$$

Base case. When $m = 2$:

$$\begin{aligned} b_{\sigma_1 \vee \sigma_2} &= b_{\sigma_2} + b_{\sigma_1} - b_{\sigma_1} b_{\sigma_2} \\ &= 1 - (1 - b_{\sigma_2} - b_{\sigma_1} + b_{\sigma_1} b_{\sigma_2}) \\ &= 1 - (1 - b_{\sigma_1})(1 - b_{\sigma_2}) \\ &= 1 - \prod_{i=1}^2 (1 - b_{\sigma_i}) \end{aligned}$$

Induction step. Let $k \in \mathbb{Z}^+$ be given and suppose that Lemma A.10 is true for $m = k$. Then

$$\begin{aligned} b_{\vee \{\sigma_i\}_{i \in [1..k+1]}} &= b_{\vee \{\{\sigma_i\}_{i \in [1..k]}\} \vee \sigma_{k+1}} \\ &= b_{\{\sigma_i\}_{i \in [1..k]}} + b_{\sigma_{k+1}} - b_{\{\sigma_i\}_{i \in [1..k]}} b_{\sigma_{k+1}} \\ &= \left[1 - \prod_{i=1}^k (1 - b_{\sigma_i}) \right] + b_{\sigma_{k+1}} - \left[1 - \prod_{i=1}^k (1 - b_{\sigma_i}) \right] b_{\sigma_{k+1}} \\ &= 1 - \prod_{i=1}^k (1 - b_{\sigma_i}) + b_{\sigma_{k+1}} - b_{\sigma_{k+1}} + b_{\sigma_{k+1}} \prod_{i=1}^k (1 - b_{\sigma_i}) \\ &= 1 - \prod_{i=1}^k (1 - b_{\sigma_i}) + b_{\sigma_{k+1}} \prod_{i=1}^k (1 - b_{\sigma_i}) \\ &= 1 - \left[\prod_{i=1}^k (1 - b_{\sigma_i}) \right] [1 - b_{\sigma_{k+1}}] \\ &= 1 - \prod_{i=1}^{k+1} (1 - b_{\sigma_i}) \end{aligned}$$

Thus, Lemma A.10 holds for $m = k + 1$. By the principle of induction, Lemma A.10 is true for all $m \in \mathbb{Z}^+$. \square

2. The mathematical proof of the relation d_α :

Lemma A.11. $d_{\alpha=\{\sigma_1, \dots, \sigma_m\}} = d_{\vee\{\sigma_i\}} = \prod_{i=1}^m d_{\sigma_i}$

Proof. We prove by induction that, for all $m \in \mathbb{Z}^+$,

$$d_{\vee\{\sigma_i\}_{i \in [1..m]}} = \prod_{i=1}^m d_{\sigma_i}$$

Base case. When $m = 2$:

$$d_{\sigma_1 \vee \sigma_2} = d_{\sigma_1} d_{\sigma_2} = \prod_{i=1}^2 d_{\sigma_i}$$

Induction step. Let $k \in \mathbb{Z}^+$ be given and suppose that Lemma A.11 is true for $m = k$. Then

$$\begin{aligned} d_{\vee\{\sigma_i\}_{i \in [1..k+1]}} &= d_{\vee\{\{\sigma_i\}_{i \in [1..k]}\} \vee \sigma_{k+1}} = d_{\vee\{\{\sigma_i\}_{i \in [1..k]}\}} d_{\sigma_{k+1}} \\ &= \prod_{i=1}^k d_{\sigma_i} d_{\sigma_{k+1}} = \prod_{i=1}^{k+1} d_{\sigma_i} \end{aligned}$$

Thus, Lemma A.11 holds for $m = k + 1$. By the principle of induction, Lemma A.11 is true for all $m \in \mathbb{Z}^+$. \square

3. The mathematical proof of the relation u_α :

Lemma A.12. $u_{\alpha=\{\sigma_1, \dots, \sigma_m\}} = u_{\vee\{\sigma_i\}} = \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})$

Proof. We prove by induction that, for all $m \in \mathbb{Z}^+$,

$$u_{\vee\{\sigma_i\}_{i \in [1..m]}} = \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})$$

Base case. When $m = 2$:

$$\begin{aligned} u_{\sigma_1 \vee \sigma_2} &= d_{\sigma_1} u_{\sigma_2} + u_{\sigma_1} d_{\sigma_2} + u_{\sigma_1} u_{\sigma_2} \\ &= d_{\sigma_1} u_{\sigma_2} + u_{\sigma_1} d_{\sigma_2} + u_{\sigma_1} u_{\sigma_2} + d_{\sigma_1} d_{\sigma_2} - d_{\sigma_1} d_{\sigma_2} \\ &= (d_{\sigma_1} + u_{\sigma_1})(d_{\sigma_2} + u_{\sigma_2}) - d_{\sigma_1} d_{\sigma_2} \\ &= \prod_{i=1}^2 (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^2 (d_{\sigma_i}) \end{aligned}$$

Induction step. Let $k \in \mathbb{Z}^+$ be given and suppose that Lemma A.12

is true for $m = k$. Then

$$\begin{aligned}
u_{\vee\{\sigma_i\}_{i \in [1..k+1]}} &= u_{\vee\{\{\sigma_i\}_{i \in [1..k]}\} \vee \sigma_{k+1}} \\
&= d_{\{\sigma_i\}_{i \in [1..k]}} u_{\sigma_{k+1}} + u_{\{\sigma_i\}_{i \in [1..k]}} d_{\sigma_{k+1}} + u_{\{\sigma_i\}_{i \in [1..k]}} u_{\sigma_{k+1}} \\
&= \left[\prod_{i=1}^k d_{\sigma_i} \right] u_{\sigma_{k+1}} + \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] d_{\sigma_{k+1}} \\
&\quad + \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] u_{\sigma_{k+1}} \\
&= \prod_{i=1}^k d_{\sigma_i} u_{\sigma_{k+1}} + \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) d_{\sigma_{k+1}} - \prod_{i=1}^k (d_{\sigma_i}) d_{\sigma_{k+1}} \\
&\quad + \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) u_{\sigma_{k+1}} - \prod_{i=1}^k (d_{\sigma_i}) u_{\sigma_{k+1}} \\
&= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) \right] (d_{\sigma_{k+1}} + u_{\sigma_{k+1}}) - \prod_{i=1}^{k+1} (d_{\sigma_i}) \\
&= \prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^{k+1} (d_{\sigma_i})
\end{aligned}$$

Thus, Lemma A.12 holds for $m = k + 1$. By the principle of induction, Lemma A.12 is true for all $m \in \mathbb{Z}^+$. \square

4. The mathematical proof of the relation a_α :

Lemma A.13.
$$a_\alpha = \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})}$$

Proof. We prove by induction that, for all $m \in \mathbb{Z}^+$,

$$a_{\vee\{\sigma_i\}} = \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})}$$

Base case. When $m = 2$:

$$a_{\sigma_1 \vee \sigma_2} = \frac{u_{\sigma_1} a_{\sigma_1} + u_{\sigma_2} a_{\sigma_2} - b_{\sigma_1} u_{\sigma_2} a_{\sigma_2} - b_{\sigma_2} u_{\sigma_1} a_{\sigma_1} - u_{\sigma_1} a_{\sigma_1} u_{\sigma_2} a_{\sigma_2}}{u_{\sigma_1} + u_{\sigma_2} - b_{\sigma_1} u_{\sigma_2} - b_{\sigma_2} u_{\sigma_1} - u_{\sigma_1} u_{\sigma_2}}$$

We denote the numerator with γ , and the denominator with β . Thus, we have the following.

$$a_{\sigma_1 \vee \sigma_2} = \frac{\gamma}{\beta}$$

$$\begin{aligned}
\gamma &= u_{\sigma_1} a_{\sigma_1} + u_{\sigma_2} a_{\sigma_2} - b_{\sigma_1} u_{\sigma_2} a_{\sigma_2} - b_{\sigma_2} u_{\sigma_1} a_{\sigma_1} - u_{\sigma_1} a_{\sigma_1} u_{\sigma_2} a_{\sigma_2} \\
&= (1 - b_{\sigma_1} - b_{\sigma_2} + b_{\sigma_1} b_{\sigma_2}) - (1 - b_{\sigma_1} - b_{\sigma_2} + b_{\sigma_1} b_{\sigma_2}) \\
&\quad + u_{\sigma_1} a_{\sigma_1} + u_{\sigma_2} a_{\sigma_2} - b_{\sigma_1} u_{\sigma_2} a_{\sigma_2} - b_{\sigma_2} u_{\sigma_1} a_{\sigma_1} - u_{\sigma_1} a_{\sigma_1} u_{\sigma_2} a_{\sigma_2} \\
&= (1 - b_{\sigma_1})(1 - b_{\sigma_2}) - (1 - b_{\sigma_1} - u_{\sigma_1} a_{\sigma_1} - b_{\sigma_2} + b_{\sigma_1} b_{\sigma_2} + b_{\sigma_2} u_{\sigma_1} a_{\sigma_1} \\
&\quad - u_{\sigma_2} a_{\sigma_2} + b_{\sigma_1} u_{\sigma_2} a_{\sigma_2} + u_{\sigma_1} a_{\sigma_1} u_{\sigma_2} a_{\sigma_2}) \\
&= (1 - b_{\sigma_1})(1 - b_{\sigma_2}) \\
&\quad - [(1 - b_{\sigma_1} - u_{\sigma_1} a_{\sigma_1}) - b_{\sigma_2}(1 - b_{\sigma_1} - u_{\sigma_1} a_{\sigma_1}) - u_{\sigma_2} a_{\sigma_2}(1 - b_{\sigma_1} - u_{\sigma_1} a_{\sigma_1})] \\
&= (1 - b_{\sigma_1})(1 - b_{\sigma_2}) - [(1 - b_{\sigma_1} - u_{\sigma_1} a_{\sigma_1})(1 - b_{\sigma_2} - u_{\sigma_2} a_{\sigma_2})] \\
&= (d_{\sigma_1} + u_{\sigma_1})(d_{\sigma_2} + u_{\sigma_2}) - [(d_{\sigma_1} + u_{\sigma_1} - u_{\sigma_1} a_{\sigma_1})(d_{\sigma_2} + u_{\sigma_2} - u_{\sigma_2} a_{\sigma_2})] \\
&= \prod_{i=1}^2 (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^2 (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})
\end{aligned}$$

$$\begin{aligned}
\beta &= u_{\sigma_1} + u_{\sigma_2} - b_{\sigma_1} u_{\sigma_2} - b_{\sigma_2} u_{\sigma_1} - u_{\sigma_1} u_{\sigma_2} \\
&= u_{\sigma_1} + u_{\sigma_2} - (1 - d_{\sigma_1} - u_{\sigma_1}) u_{\sigma_2} - (1 - d_{\sigma_2} - u_{\sigma_2}) u_{\sigma_1} - u_{\sigma_1} u_{\sigma_2} \\
&= d_{\sigma_1} u_{\sigma_2} + u_{\sigma_1} u_{\sigma_2} + d_{\sigma_2} u_{\sigma_1} + u_{\sigma_2} u_{\sigma_1} - u_{\sigma_1} u_{\sigma_2} \\
&= d_{\sigma_1} u_{\sigma_2} + d_{\sigma_2} u_{\sigma_1} + u_{\sigma_2} u_{\sigma_1} \\
&= u_{\sigma_1 \wedge \sigma_2} \\
&= \prod_{i=1}^2 (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^2 (d_{\sigma_i})
\end{aligned}$$

$$a_{\sigma_1 \vee \sigma_2} = \frac{\gamma}{\beta} = \frac{\prod_{i=1}^2 (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^2 (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^2 (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^2 (d_{\sigma_i})}$$

Induction step. Let $k \in \mathbb{Z}^+$ be given and suppose that Lemma A.13 is true for $m = k$. Then

$$\begin{aligned}
a_{\vee\{\sigma_i\}_{i \in [1..k+1]}} &= a_{\vee\{\{\sigma_i\}_{i \in [1..k]}\} \vee \sigma_{k+1}} = \frac{\gamma}{\beta} : \\
\gamma &= u_{\{\sigma_i\}_{i \in [1..k]}} a_{\{\sigma_i\}_{i \in [1..k]}} + u_{\sigma_{k+1}} a_{\sigma_{k+1}} - b_{\{\sigma_i\}_{i \in [1..k]}} u_{\sigma_{k+1}} a_{\sigma_{k+1}} \\
&\quad - b_{\sigma_{k+1}} u_{\{\sigma_i\}_{i \in [1..k]}} a_{\{\sigma_i\}_{i \in [1..k]}} - u_{\{\sigma_i\}_{i \in [1..k]}} a_{\{\sigma_i\}_{i \in [1..k]}} u_{\sigma_{k+1}} a_{\sigma_{k+1}}, \\
\beta &= u_{\{\sigma_i\}_{i \in [1..k]}} + u_{\sigma_{k+1}} - b_{\{\sigma_i\}_{i \in [1..k]}} u_{\sigma_{k+1}} - b_{\sigma_{k+1}} u_{\{\sigma_i\}_{i \in [1..k]}} - u_{\{\sigma_i\}_{i \in [1..k]}} u_{\sigma_{k+1}}
\end{aligned}$$

Now, we compute γ and β .

$$\begin{aligned}
\gamma &= u_{\alpha} a_{\alpha} + u_{\sigma_{k+1}} a_{\sigma_{k+1}} - b_{\alpha} u_{\sigma_{k+1}} a_{\sigma_{k+1}} - b_{\sigma_{k+1}} u_{\alpha} a_{\alpha} - u_{\alpha} a_{\alpha} u_{\sigma_{k+1}} a_{\sigma_{k+1}} \\
&= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] \left[\frac{\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i})} \right] \\
&\quad + u_{\sigma_{k+1}} a_{\sigma_{k+1}} - \left[1 - \prod_{i=1}^k (1 - b_{\sigma_i}) \right] u_{\sigma_{k+1}} a_{\sigma_{k+1}} \\
&\quad - b_{\sigma_{k+1}} \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] \left[\frac{\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i})} \right] \\
&\quad - u_{\sigma_{k+1}} a_{\sigma_{k+1}} \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] \left[\frac{\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i})} \right]
\end{aligned}$$

$$\begin{aligned}
\gamma &= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right] \\
&+ u_{\sigma_{k+1}} a_{\sigma_{k+1}} - \left[1 - \prod_{i=1}^k (1 - b_{\sigma_i}) \right] u_{\sigma_{k+1}} a_{\sigma_{k+1}} \\
&- b_{\sigma_{k+1}} \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right] \\
&- u_{\sigma_{k+1}} a_{\sigma_{k+1}} \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right] \\
\gamma &= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right] [1 - b_{\sigma_{k+1}} - u_{\sigma_{k+1}} a_{\sigma_{k+1}}] \\
&+ u_{\sigma_{k+1}} a_{\sigma_{k+1}} \left[1 - 1 + \prod_{i=1}^k (1 - b_{\sigma_i}) \right] \\
\gamma &= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right] [d_{\sigma_{k+1}} + u_{\sigma_{k+1}} - u_{\sigma_{k+1}} a_{\sigma_{k+1}}] \\
&+ u_{\sigma_{k+1}} a_{\sigma_{k+1}} \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) \right] \\
\gamma &= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) \right] [d_{\sigma_{k+1}} + u_{\sigma_{k+1}} - u_{\sigma_{k+1}} a_{\sigma_{k+1}}] - \left[\prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right] \\
&+ \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) \right] u_{\sigma_{k+1}} a_{\sigma_{k+1}} \\
\gamma &= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) \right] [d_{\sigma_{k+1}} + u_{\sigma_{k+1}}] - \left[\prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right] \\
\gamma &= \left[\prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i}) \right] - \left[\prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right]
\end{aligned}$$

$$\begin{aligned}
\beta &= u_{\alpha} + u_{\sigma_{k+1}} - b_{\alpha} u_{\sigma_{k+1}} - b_{\sigma_{k+1}} u_{\alpha} - u_{\alpha} u_{\sigma_{k+1}} \\
\beta &= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] + u_{\sigma_{k+1}} - \left[1 - \prod_{i=1}^k (1 - b_{\sigma_i}) \right] u_{\sigma_{k+1}} \\
&- b_{\sigma_{k+1}} \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] - \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] u_{\sigma_{k+1}} \\
\beta &= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] + u_{\sigma_{k+1}} - u_{\sigma_{k+1}} + \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) u_{\sigma_{k+1}} \\
&- [1 - d_{\sigma_{k+1}} - u_{\sigma_{k+1}}] \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] - \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) u_{\sigma_{k+1}} + \prod_{i=1}^k (d_{\sigma_i}) u_{\sigma_{k+1}} \\
\beta &= \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] \\
&- [1 - d_{\sigma_{k+1}} - u_{\sigma_{k+1}}] \left[\prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \right] + \prod_{i=1}^k (d_{\sigma_i}) u_{\sigma_{k+1}} \\
\beta &= \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^k (d_{\sigma_i}) \\
&- [1 - d_{\sigma_{k+1}} - u_{\sigma_{k+1}}] \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) + [1 - d_{\sigma_{k+1}} - u_{\sigma_{k+1}}] \prod_{i=1}^k (d_{\sigma_i}) + \prod_{i=1}^k (d_{\sigma_i}) u_{\sigma_{k+1}} \\
\beta &= \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) [1 - 1 + d_{\sigma_{k+1}} + u_{\sigma_{k+1}}] - \prod_{i=1}^k (d_{\sigma_i}) [1 - 1 + d_{\sigma_{k+1}} + u_{\sigma_{k+1}} - u_{\sigma_{k+1}}] \\
&= \prod_{i=1}^k (d_{\sigma_i} + u_{\sigma_i}) [d_{\sigma_{k+1}} + u_{\sigma_{k+1}}] - \prod_{i=1}^k (d_{\sigma_i}) [d_{\sigma_{k+1}}] \\
&= \prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^{k+1} (d_{\sigma_i})
\end{aligned}$$

$$a_{\vee\{\sigma_i\}_{i \in [1..k+1]}} = \frac{\gamma}{\beta} = \frac{\left[\prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i}) \right] - \left[\prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) \right]}{\prod_{i=1}^{k+1} (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^{k+1} (d_{\sigma_i})}$$

Thus, Lemma A.13 holds for $m = k + 1$. By the principle of induction, Lemma A.13 is true for all $m \in \mathbb{Z}^+$. \square

A.1.2.1 Verifications

In this section we verify that the previous relations hold the properties of the opinion parameters in subjective logic (*i.e.*, $b + d + u = 1$ and $0 < b, d, u, a < 1$) so we verify the following:

$$\begin{cases} 0 < b_\alpha = 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) < 1 \\ 0 < d_\alpha = \prod_{i=1}^m d_{\sigma_i} < 1 \\ 0 < u_\alpha = \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) < 1 \\ 0 < a_\alpha = \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})} < 1 \\ b_\alpha + d_\alpha + u_\alpha = 1 \end{cases}$$

Lemma A.14. $0 < b_\alpha = 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) < 1$

Proof.

$$\begin{aligned} \forall i \in [1..m] : \\ 0 < b_{\sigma_i} < 1 &\Rightarrow \\ 1 > 1 - b_{\sigma_i} > 0 &\Rightarrow \\ 1 > \prod_{i=1}^m (1 - b_{\sigma_i}) > 0 &\Rightarrow \\ 0 < 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) < 1 &\Rightarrow \\ 0 < b_\alpha < 1 \end{aligned}$$

\square

Lemma A.15. $0 < d_\alpha = \prod_{i=1}^m d_{\sigma_i} < 1$

Proof.

$$\forall i \in [1..m] : 0 < d_{\sigma_i} < 1$$

The multiplication of several values between 0 and 1 is between 0 and 1 \Rightarrow

$$0 < \prod_{i=1}^m d_{\sigma_i} < 1 \Rightarrow 0 < d_\alpha < 1$$

\square

Lemma A.16. $0 < u_\alpha = \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) < 1$

Proof. Considering the left side of the relation, we have:

$$\begin{aligned}
& \forall i \in [1..m] : u_{\sigma_i} > 0 \Rightarrow \\
& d_{\sigma_i} + u_{\sigma_i} > d_{\sigma_i} \Rightarrow \\
& \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) > \prod_{i=1}^m (d_{\sigma_i}) \Rightarrow \\
& \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) > 0 \Rightarrow \\
& u_{\alpha} > 0
\end{aligned}$$

Considering the right side of the relation, we have:

$$\begin{aligned}
& \forall i \in [1..m] : d_{\sigma_i} + u_{\sigma_i} < 1 \Rightarrow \\
& \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) < 1 \Rightarrow \\
& \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) < 1 \Rightarrow \\
& u_{\alpha} < 1
\end{aligned}$$

□

Lemma A.17. $0 < a_{\alpha} = \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})} < 1$

Proof. Considering the left side of the relation, we have:

$$\left. \begin{aligned}
& \forall i \in [1..m] : \\
& d_{\sigma_i} + u_{\sigma_i} > d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i} \Rightarrow \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) > 0 \\
& d_{\sigma_i} + u_{\sigma_i} > d_{\sigma_i} \Rightarrow \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) > 0
\end{aligned} \right\} \Rightarrow$$

$$a_{\alpha} = \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})} > 0$$

Considering the right side of the relation, we have:

$$\begin{aligned}
& \forall i \in [1..m] : 0 < a_{\sigma_i} < 1 \Rightarrow \\
& u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i} > 0 \Rightarrow \\
& d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i} > d_{\sigma_i} \Rightarrow \\
& \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) > \prod_{i=1}^m (d_{\sigma_i}) \Rightarrow \\
& - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) < - \prod_{i=1}^m (d_{\sigma_i}) \Rightarrow \\
& \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i}) < \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) \Rightarrow \\
& \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})} < 1
\end{aligned}$$

□

Lemma A.18. $b_{\alpha} + d_{\alpha} + u_{\alpha} = 1$

Proof.

$$\left\{ \begin{array}{l}
b_{\alpha} = 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) \\
d_{\alpha} = \prod_{i=1}^m d_{\sigma_i} \\
u_{\alpha} = \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})
\end{array} \right. \Rightarrow$$

$$\begin{aligned}
b_\alpha + d_\alpha + u_\alpha &= 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) + \prod_{i=1}^m d_{\sigma_i} + \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) \\
b_\alpha + d_\alpha + u_\alpha &= 1 - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) + \prod_{i=1}^m d_{\sigma_i} + \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) \\
b_\alpha + d_\alpha + u_\alpha &= 1
\end{aligned}$$

□

A.1.3 Split Relations (mathematical proof)

In this section, we prove the Relations 6.9, 6.10 of the method **Split** presented in Section 6.2.3.2.

A.1.3.1 Relations 6.9: splitting a dependent opinion into two independent opinions (mathematical proof):

$$\begin{aligned}
&\left\{ \begin{array}{l} b_{A_1} \vee b_{A_2} = b_A \\ d_{A_1} \vee d_{A_2} = d_A \\ u_{A_1} \vee u_{A_2} = u_A \\ a_{A_1} \vee a_{A_2} = a_A \end{array} \right\} \wedge \left\{ \begin{array}{l} b_{A_1} = b_{A_2} \\ d_{A_1} = d_{A_2} \\ u_{A_1} = u_{A_2} \\ a_{A_1} = a_{A_2} \end{array} \right\} \Rightarrow \\
&\left\{ \begin{array}{l} b_{A_1} = b_{A_2} = 1 - \sqrt{1 - b_A} \\ d_{A_1} = d_{A_2} = \sqrt{d_A} \\ u_{A_1} = u_{A_2} = \frac{\sqrt{d_A + u_A} - \sqrt{d_A}}{2} \\ a_{A_1} = a_{A_2} = \frac{\sqrt{1 - b_A} - \sqrt{1 - b_A - a_A u_A}}{\sqrt{d_A + u_A} - \sqrt{d_A}} \end{array} \right.
\end{aligned}$$

In the following, we denote $O_A = (b_A, d_A, u_A, a_A)$ with $O_A = (b, d, u, a)$ for simplicity.

1. The mathematical proof of the relation b :

Lemma A.19. $(b_{A_1} \vee b_{A_2} = b) \wedge (b_{A_1} = b_{A_2}) \Rightarrow b_{A_1} = b_{A_2} = 1 - \sqrt{1 - b}$

Proof.

$$\begin{aligned}
b_{A_1} \vee b_{A_2} = b &\Rightarrow \\
b_{A_1} + b_{A_2} - b_{A_1} b_{A_2} = b &\Rightarrow \\
b_{A_1} + b_{A_1} - b_{A_1} b_{A_1} = b &\Rightarrow \\
b_{A_1}^2 - 2b_{A_1} + b = 0
\end{aligned}$$

This is an equation from the second degree

$$\Delta = 4 - 4b = 4(1 - b) > 0 \Rightarrow$$

$$b_{A_1} = b_{A_2} = \frac{2 + 2\sqrt{(1 - b)}}{2} = 1 + \sqrt{(1 - b)} > 1 \quad (\text{refused solution})$$

$$b_{A_1} = b_{A_2} = \frac{2 - 2\sqrt{(1 - b)}}{2} = 1 - \sqrt{(1 - b)}$$

$$0 < b_{A_1} = b_{A_2} = 1 - \sqrt{(1 - b)} < 1 \quad (\text{accepted solution})$$

□

2. The mathematical proof of the relation d :

Lemma A.20. $(d_{A1} \vee d_{A2} = d) \wedge (d_{A1} = d_{A2}) \Rightarrow d_{A1} = d_{A2} = \sqrt{d}$

Proof.

$$\begin{aligned} d_{A1} \vee d_{A2} = d &\Rightarrow \\ d_{A1}d_{A2} = d &\Rightarrow \\ d_{A1}^2 = d &\Rightarrow \\ d_{A1} = d_{A2} = -\sqrt{d} < 0 &\quad (\text{refused solution}) \\ d_{A1} = d_{A2} = \sqrt{d} & \\ 0 < d_{A1} = d_{A2} = \sqrt{d} < 1 &\quad (\text{accepted solution}) \end{aligned}$$

□

3. The mathematical proof of the relation u :

Lemma A.21. $(u_{A1} \vee u_{A2} = u) \wedge (u_{A1} = u_{A2}) \Rightarrow u_{A1} = u_{A2} = \sqrt{d+u} - \sqrt{d}$

Proof.

$$\begin{aligned} u_{A1} \vee u_{A2} = u &\Rightarrow \\ d_{A1}u_{A2} + d_{A2}u_{A1} + u_{A1}u_{A2} = u &\Rightarrow \\ d_{A1}u_{A1} + d_{A1}u_{A1} + u_{A1}u_{A1} = u &\Rightarrow \\ 2d_{A1}u_{A1} + u_{A1}^2 = u &\Rightarrow \\ u_{A1}^2 + 2d_{A1}u_{A1} - u = 0 &\Rightarrow \\ u_{A1}^2 + 2\sqrt{d}u_{A1} - u = 0 &\Rightarrow \end{aligned}$$

This is an equation from the second degree

$$\begin{aligned} \Delta = 4d + 4u = 4(d+u) > 0 &\Rightarrow \\ u_{A1} = u_{A2} = \frac{-2\sqrt{d} - 2\sqrt{(d+u)}}{2} = -\sqrt{d} - \sqrt{(d+u)} < 0 & \\ (\text{refused solution}) & \\ u_{A1} = u_{A2} = \frac{-2\sqrt{d} + 2\sqrt{(d+u)}}{2} = -\sqrt{d} + \sqrt{(d+u)} & \\ 0 < u_{A1} = u_{A2} = -\sqrt{d} + \sqrt{(d+u)} < 1 &\quad (\text{accepted solution}) \end{aligned}$$

□

4. The mathematical proof of the relation a :

Lemma A.22. $(a_{A1} \vee a_{A2} = a) \wedge (a_{A1} = a_{A2}) \Rightarrow a_{A1} = a_{A2} = \frac{\sqrt{1-b} - \sqrt{1-b-au}}{\sqrt{d+u} - \sqrt{d}}$

Proof.

$$\begin{aligned}
a_{A1} \vee a_{A2} = a &\Rightarrow \\
\frac{u_{A1}a_{A1} + u_{A2}a_{A2} - b_{A1}u_{A2}a_{A2} - b_{A2}u_{A1}a_{A1} - u_{A1}a_{A1}u_{A2}a_{A2}}{u_{A1} + u_{A2} - b_{A1}u_{A2} - b_{A2}u_{A1} - u_{A1}u_{A2}} = a &\Rightarrow \\
\frac{u_{A1}a_{A1} + u_{A1}a_{A1} - b_{A1}u_{A1}a_{A1} - b_{A1}u_{A1}a_{A1} - u_{A1}a_{A1}u_{A1}a_{A1}}{u_{A1} + u_{A1} - b_{A1}u_{A1} - b_{A1}u_{A1} - u_{A1}u_{A1}} = a &\Rightarrow \\
\frac{[2 - 2b_{A1}]u_{A1}a_{A1} - u_{A1}^2 a_{A1}^2}{2u_{A1} - 2b_{A1}u_{A1} - u_{A1}^2} = a &\Rightarrow \\
\frac{2[1 - (1 - \sqrt{1-b})][-\sqrt{d} + \sqrt{(d+u)}]a_{A1} - [-\sqrt{d} + \sqrt{(d+u)}]^2 a_{A1}^2}{2[-\sqrt{d} + \sqrt{(d+u)}] - 2[1 - \sqrt{1-b})][-\sqrt{d} + \sqrt{(d+u)}] - [-\sqrt{d} + \sqrt{(d+u)}]^2} = a &\Rightarrow \\
\frac{2[\sqrt{(d+u)}][-\sqrt{d} + \sqrt{(d+u)}]a_{A1} - [-\sqrt{d} + \sqrt{(d+u)}]^2 a_{A1}^2}{[-\sqrt{d} + \sqrt{(d+u)}][2 - 2[1 - \sqrt{(d+u)}] - [-\sqrt{d} + \sqrt{(d+u)}]]} = a &\Rightarrow \\
\frac{2[\sqrt{(d+u)}]a_{A1} - [-\sqrt{d} + \sqrt{(d+u)}]a_{A1}^2}{[2 - 2[1 - \sqrt{(d+u)}] - [-\sqrt{d} + \sqrt{(d+u)}]]} = a &\Rightarrow \\
\frac{[\sqrt{d} - \sqrt{(d+u)}]a_{A1}^2 + 2[\sqrt{(d+u)}]a_{A1}}{[2\sqrt{(d+u)}] + [\sqrt{d} - \sqrt{(d+u)}]} = a &\Rightarrow \\
[\sqrt{d} - \sqrt{(d+u)}]a_{A1}^2 + 2[\sqrt{(d+u)}]a_{A1} = [\sqrt{(d+u)} + \sqrt{d}]a &\Rightarrow \\
[\sqrt{d} - \sqrt{(d+u)}]a_{A1}^2 + 2[\sqrt{(d+u)}]a_{A1} - [\sqrt{(d+u)} + \sqrt{d}]a = 0 &\Rightarrow
\end{aligned}$$

This is an equation from the second degree

$$\begin{aligned}
\Delta &= 4(d+u) - 4[\sqrt{d} - \sqrt{(d+u)}][-\sqrt{(d+u)} - \sqrt{d}]a \\
\Delta &= 4d + 4u + 4[d - d - u]a \\
\Delta &= 4[d + u - ua] \\
\Delta &= 4[1 - b - ua] \\
a_{A1} = a_{A2} &= \frac{-2[\sqrt{(d+u)}] + 2\sqrt{(1-b-ua)}}{2[\sqrt{d} - \sqrt{(d+u)}]} \\
&= \frac{[\sqrt{(1-b)}] + \sqrt{(1-b-ua)}}{[\sqrt{(d+u)} - \sqrt{d}]} > 1 \text{ (refused solution)} \\
a_{A1} = a_{A2} &= \frac{-2[\sqrt{(d+u)}] - 2\sqrt{(1-b-ua)}}{2[\sqrt{d} - \sqrt{(d+u)}]} \\
&= \frac{[\sqrt{(1-b)}] - \sqrt{(1-b-ua)}}{[\sqrt{(d+u)} - \sqrt{d}]} \\
0 < a_{A1} = a_{A2} &= \frac{[\sqrt{(1-b)}] - \sqrt{(1-b-ua)}}{[\sqrt{(d+u)} - \sqrt{d}]} < 1 \text{ (accepted solution)}
\end{aligned}$$

□

A.1.3.2 Relations 6.10: splitting a dependent opinion into n independent opinions (mathematical proof):

Relations 6.10 can be proved by induction. In this section, we limit ourselves on verifying that the obtained Relations 6.10 are the appropriate

ones by proving the following:

$$\left\{ \begin{array}{l} b_{A_1} \vee b_{A_2} \vee \dots \vee b_{A_n} = b_A \\ 0 \leq b_{A_i} = 1 - (1 - b_A)^{\frac{1}{n}} \leq 1 \\ d_{A_1} \vee d_{A_2} \vee \dots \vee d_{A_n} = d_A \\ 0 \leq d_{A_i} = d_A^{\frac{1}{n}} \leq 1 \\ u_{A_1} \vee u_{A_2} \vee \dots \vee u_{A_n} = u_A \\ 0 \leq u_{A_i} = (d_A + u_A)^{\frac{1}{n}} - d_A^{\frac{1}{n}} \leq 1 \\ a_{A_1} \vee a_{A_2} \vee \dots \vee a_{A_n} = a_A \\ 0 \leq a_{A_i} = \frac{(1-b_A)^{\frac{1}{n}} - (1-b_A - a_A u_A)^{\frac{1}{n}}}{(d_A + u_A)^{\frac{1}{n}} - d_A^{\frac{1}{n}}} \leq 1 \\ b_{A_i} + d_{A_i} + u_{A_i} = 1 \end{array} \right.$$

In the following, we denote $O_A = (b_A, d_A, u_A, a_A)$ with $O_A = (b, d, u, a)$ for simplicity.

Lemma A.23. $b_{A_1} \vee b_{A_2} \vee \dots \vee b_{A_n} = b$

Proof.

$$\left\{ \begin{array}{l} \text{From Relations 6.8} \Rightarrow b_{\vee\{A_i\}} = 1 - \prod_{i=1}^n (1 - b_{A_i}) \\ b_{A_i} = 1 - (1 - b)^{\frac{1}{n}} \end{array} \right. \Rightarrow$$

$$\begin{aligned} b_{\vee\{A_i\}} &= 1 - \prod_{i=1}^n (1 - (1 - (1 - b)^{\frac{1}{n}})) \\ &= 1 - \prod_{i=1}^n ((1 - b)^{\frac{1}{n}}) \\ &= 1 - (1 - b) \\ &= b \end{aligned}$$

□

Lemma A.24. $0 \leq 1 - (1 - b)^{\frac{1}{n}} \leq 1$

Proof.

$$\begin{aligned} 0 \leq b \leq 1 &\Rightarrow \\ 0 \leq 1 - b \leq 1 &\Rightarrow \\ 0 \leq (1 - b)^{\frac{1}{n}} \leq 1 &\Rightarrow \\ 0 \leq 1 - (1 - b)^{\frac{1}{n}} \leq 1 & \end{aligned}$$

□

Lemma A.25. $d_{A_1} \vee d_{A_2} \vee \dots \vee d_{A_n} = d$

Proof.

$$\left\{ \begin{array}{l} \text{From Relations 6.8} \Rightarrow d_{\vee\{A_i\}} = \prod_{i=1}^n d_{A_i} \\ d_{A_i} = d^{\frac{1}{n}} \end{array} \right. \Rightarrow$$

$$\begin{aligned} d_{\vee\{A_i\}} &= \prod_{i=1}^n d^{\frac{1}{n}} \\ &= d \end{aligned}$$

□

Lemma A.26. $0 \leq d^{\frac{1}{n}} \leq 1$

Proof.

$$\begin{aligned} 0 \leq d \leq 1 &\Rightarrow \\ 0 \leq d^{\frac{1}{n}} &\leq 1 \end{aligned}$$

□

Lemma A.27. $u_{A_1} \vee u_{A_2} \vee \dots \vee u_{A_n} = u$

Proof.

$$\left\{ \begin{array}{l} \text{From Relations 6.8} \Rightarrow u_{\vee\{A_i\}} = \prod_{i=1}^n (d_{A_i} + u_{A_i}) - \prod_{i=1}^n (d_{A_i}) \\ d_{A_i} = d^{\frac{1}{n}} \\ u_{A_i} = (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} \end{array} \right. \Rightarrow$$

$$\begin{aligned} u_{\vee\{A_i\}} &= \prod_{i=1}^n [(d^{\frac{1}{n}}) + ((d + u)^{\frac{1}{n}} - d^{\frac{1}{n}})] - \prod_{i=1}^n d^{\frac{1}{n}} \\ &= \prod_{i=1}^n (d + u)^{\frac{1}{n}} - \prod_{i=1}^n d^{\frac{1}{n}} \\ &= d + u - d \\ &= u \end{aligned}$$

□

Lemma A.28. $0 \leq (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} \leq 1$

Proof. Considering the left side of the relation, we have:

$$\begin{aligned} d &\leq d + u \Rightarrow \\ d^{\frac{1}{n}} &\leq (d + u)^{\frac{1}{n}} \Rightarrow \\ 0 &\leq (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} \end{aligned}$$

Considering the right side of the relation, we have:

$$d + u = 1 - b$$

$$0 \leq b \leq 1 \Rightarrow$$

$$\begin{aligned} d + u &\leq 1 \Rightarrow \\ (d + u)^{\frac{1}{n}} &\leq 1 \Rightarrow \\ (d + u)^{\frac{1}{n}} &\leq 1 + d^{\frac{1}{n}} \Rightarrow \\ (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} &\leq 1 \Rightarrow \\ u &\leq 1 \end{aligned}$$

□

Lemma A.29. $a_{A_1} \vee a_{A_2} \vee \dots \vee a_{A_n} = a$

Proof.

$$\left\{ \begin{array}{l} \text{From Relations 6.8} \Rightarrow a_{\vee\{A_i\}} = \frac{\prod_{i=1}^n (d_{A_i} + u_{A_i}) - \prod_{i=1}^n (d_{A_i} + u_{A_i} - u_{A_i} a_{A_i})}{\prod_{i=1}^n (d_{A_i} + u_{A_i}) - \prod_{i=1}^n (d_{A_i})} \\ b_{A_i} = 1 - (1 - b)^{\frac{1}{n}} \\ d_{A_i} = d^{\frac{1}{n}} \\ u_{A_i} = (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} \\ a_{A_i} = \frac{(1 - b)^{\frac{1}{n}} - (1 - b - au)^{\frac{1}{n}}}{(d + u)^{\frac{1}{n}} - d^{\frac{1}{n}}} \end{array} \right. \Rightarrow$$

$$a_{\vee\{A_i\}} = \frac{\prod_{i=1}^n (d^{\frac{1}{n}} + ((d + u)^{\frac{1}{n}} - d^{\frac{1}{n}})) - \prod_{i=1}^n [(d^{\frac{1}{n}} + ((d + u)^{\frac{1}{n}} - d^{\frac{1}{n}}) - ((1 - b)^{\frac{1}{n}} - (1 - b - au)^{\frac{1}{n}})]}{\prod_{i=1}^n (d^{\frac{1}{n}} + ((d + u)^{\frac{1}{n}} - d^{\frac{1}{n}})) - \prod_{i=1}^n (d^{\frac{1}{n}})}$$

$$\begin{aligned} a_{\vee\{A_i\}} &= \frac{\prod_{i=1}^n (d + u)^{\frac{1}{n}} - \prod_{i=1}^n [(d + u)^{\frac{1}{n}} - ((1 - b)^{\frac{1}{n}} - (1 - b - au)^{\frac{1}{n}})]}{\prod_{i=1}^n (d + u)^{\frac{1}{n}} - \prod_{i=1}^n (d^{\frac{1}{n}})} \\ &= \frac{\prod_{i=1}^n (d + u)^{\frac{1}{n}} - \prod_{i=1}^n [(1 - b - au)^{\frac{1}{n}}]}{\prod_{i=1}^n (d + u)^{\frac{1}{n}} - \prod_{i=1}^n (d^{\frac{1}{n}})} \\ &= \frac{\prod_{i=1}^n (1 - b)^{\frac{1}{n}} - \prod_{i=1}^n [(1 - b - au)^{\frac{1}{n}}]}{\prod_{i=1}^n (d + u)^{\frac{1}{n}} - \prod_{i=1}^n (d^{\frac{1}{n}})} \\ &= \frac{(1 - b) - (1 - b - au)}{(d + u) - (d)} \\ &= a \end{aligned}$$

□

Lemma A.30. $0 \leq a_{A_i} = \frac{(1 - b)^{\frac{1}{n}} - (1 - b - au)^{\frac{1}{n}}}{(d + u)^{\frac{1}{n}} - d^{\frac{1}{n}}} \leq 1$

Proof. Considering the left side of the relation, we have:

$$\left\{ \begin{array}{l} (1 - b)^{\frac{1}{n}} - (1 - b - au)^{\frac{1}{n}} \geq 0 \\ (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} \geq 0 \end{array} \right. \Rightarrow a_{A_i} \geq 0$$

Considering the right side of the relation, we have:

$$\begin{aligned}
d + u(1 - a) &\geq d \Rightarrow \\
(d + u - au)^{\frac{1}{n}} &\geq (d)^{\frac{1}{n}} \Rightarrow \\
(d + u)^{\frac{1}{n}} - (d + u - au)^{\frac{1}{n}} &\leq (d + u)^{\frac{1}{n}} - (d)^{\frac{1}{n}} \Rightarrow \\
(1 - b)^{\frac{1}{n}} - (1 - b - au)^{\frac{1}{n}} &\leq (d + u)^{\frac{1}{n}} - (d)^{\frac{1}{n}} \Rightarrow \\
\frac{(1 - b)^{\frac{1}{n}} - (1 - b - au)^{\frac{1}{n}}}{(d + u)^{\frac{1}{n}} - d^{\frac{1}{n}}} &\leq 1 \Rightarrow \\
a_{A_i} &\leq 1
\end{aligned}$$

□

Lemma A.31. $b_{A_i} + d_{A_i} + u_{A_i} = 1$

Proof.

$$\begin{cases} b_{A_i} = 1 - (1 - b)^{\frac{1}{n}} \\ d_{A_i} = d^{\frac{1}{n}} \\ u_{A_i} = (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} \end{cases} \Rightarrow$$

$$\begin{aligned}
b_{A_i} + d_{A_i} + u_{A_i} &= 1 - (1 - b)^{\frac{1}{n}} + d^{\frac{1}{n}} + (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} \\
&= 1 - (d + u)^{\frac{1}{n}} + d^{\frac{1}{n}} + (d + u)^{\frac{1}{n}} - d^{\frac{1}{n}} \\
&= 1
\end{aligned}$$

□

A.1.4 Comparing Copy to mTNA

In this section, we prove that **Copy** is more optimist than **mTNA**. Let m be the number of paths in a graph α that have some common nodes between the paths. If we denote the opinion about α using **Copy** by O_α^{COPY} , we have:

$$O_\alpha^{\text{COPY}} = \begin{cases} b_\alpha^{\text{COPY}} = b_{\bigvee\{\sigma_i\}} = 1 - \prod_{i=1}^m (1 - b_{\sigma_i}) \\ d_\alpha^{\text{COPY}} = d_{\bigvee\{\sigma_i\}} = \prod_{i=1}^m d_{\sigma_i} \\ u_\alpha^{\text{COPY}} = u_{\bigvee\{\sigma_i\}} = \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i}) \\ a_\alpha^{\text{COPY}} = a_{\bigvee\{\sigma_i\}} = \frac{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^m (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^m (d_{\sigma_i})} \end{cases}$$

If we denote the opinion toward α using **mTNA** by O_α^{mTNA} , we have:

$$O_\alpha^{\text{mTNA}} = \begin{cases} b_\alpha^{\text{mTNA}} = b_{\bigvee\{\sigma_i\}} = 1 - \prod_{i=1}^l (1 - b_{\sigma_i}) \\ d_\alpha^{\text{mTNA}} = d_{\bigvee\{\sigma_i\}} = \prod_{i=1}^l d_{\sigma_i} \\ u_\alpha^{\text{mTNA}} = u_{\bigvee\{\sigma_i\}} = \prod_{i=1}^l (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^l (d_{\sigma_i}) \\ a_\alpha^{\text{mTNA}} = a_{\bigvee\{\sigma_i\}} = \frac{\prod_{i=1}^l (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^l (d_{\sigma_i} + u_{\sigma_i} - u_{\sigma_i} a_{\sigma_i})}{\prod_{i=1}^l (d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^l (d_{\sigma_i})} \end{cases}$$

where $l \leq m$ because **mTNA** is based on removing the paths that have high value of uncertainty from the architecture.

In this section, we show that **Copy** is more optimist than **mTNA** by proving the following relations:

$$\begin{cases} b_\alpha^{\text{COPY}} \geq b_\alpha^{\text{MTNA}} \\ d_\alpha^{\text{COPY}} \leq d_\alpha^{\text{MTNA}} \\ \mathbb{E}(O_\alpha^{\text{COPY}}) \geq \mathbb{E}(O_\alpha^{\text{MTNA}}) \end{cases}$$

1. Comparing b_α in **Copy** and **mTNA**:

$$b_\alpha^{\text{COPY}} = 1 - \prod_{i=1}^m (1 - b_{\sigma_i})$$

$$b_\alpha^{\text{MTNA}} = 1 - \prod_{i=1}^l (1 - b_{\sigma_i})$$

Lemma A.32. $b_\alpha^{\text{MTNA}} \leq b_\alpha^{\text{COPY}}$

Proof. $m \geq l$ because **mTNA** is based on deleting uncertain paths. Thus, the number of paths in **Copy** is greater than **mTNA**. The opinions about the remaining paths are the same because in **Copy**, the duplicated nodes have the same opinion of the original node.

$$(m \geq l) \wedge (0 \leq 1 - b_{\sigma_i} \leq 1) \Rightarrow$$

$$\prod_{i=1}^m (1 - b_{\sigma_i}) \leq \prod_{i=1}^l (1 - b_{\sigma_i}) \Rightarrow$$

$$1 - \prod_{i=1}^m (1 - b_{\sigma_i}) \geq 1 - \prod_{i=1}^l (1 - b_{\sigma_i}) \Rightarrow$$

$$b_\alpha^{\text{COPY}} \geq b_\alpha^{\text{MTNA}}$$

□

2. Comparing d_α in **mTNA** and **Copy**:

$$d_\alpha^{\text{COPY}} = \prod_{i=1}^m d_{\sigma_i}$$

$$d_\alpha^{\text{MTNA}} = \prod_{i=1}^l d_{\sigma_i}$$

Lemma A.33. $d_\alpha^{\text{COPY}} \leq d_\alpha^{\text{MTNA}}$

Proof.

$$(m \geq l) \wedge (0 \leq d_{\sigma_i} \leq 1) \Rightarrow$$

$$\prod_{i=1}^m d_{\sigma_i} \leq \prod_{i=1}^l d_{\sigma_i} \Rightarrow$$

$$d_\alpha^{\text{COPY}} \leq d_\alpha^{\text{MTNA}}$$

□

3. Comparing $\mathbb{E}(O_\alpha)$ in mTNA and Copy:

$$\mathbb{E}(O_x) = b_x + a_x u_x$$

From [Jøso1]:

$$\mathbb{E}(O_x \vee y) = \mathbb{E}(O_x) + \mathbb{E}(O_y) - \mathbb{E}(O_x)\mathbb{E}(O_y) \Rightarrow$$

$$\mathbb{E}(O_\alpha) = \mathbb{E}(O_{\bigvee\{\sigma_i\}}) = 1 - \prod_{i=1}^m (1 - \mathbb{E}(O_{\sigma_i}))$$

Lemma A.34. $\mathbb{E}(O_\alpha^{\text{MTNA}}) \leq \mathbb{E}(O_\alpha^{\text{COPY}})$

Proof.

$$(m \geq l) \wedge (0 \leq 1 - \mathbb{E}(O_{\sigma_i}) \leq 1) \Rightarrow$$

$$\prod_{i=1}^m (1 - \mathbb{E}(O_{\sigma_i})) \leq \prod_{i=1}^1 (1 - \mathbb{E}(O_{\sigma_i}))$$

$$1 - \prod_{i=1}^m (1 - \mathbb{E}(O_{\sigma_i})) \geq 1 - \prod_{i=1}^1 (1 - \mathbb{E}(O_{\sigma_i})) \Rightarrow$$

$$\mathbb{E}(O_\alpha^{\text{COPY}}) \geq \mathbb{E}(O_\alpha^{\text{MTNA}})$$

□

A.1.5 Comparing Copy to Split

In this section, we show that **Copy** is more optimist than **Split** by proving the following relations:

$$\begin{cases} b_\alpha^{\text{COPY}} \geq b_\alpha^{\text{SPLIT}} \\ d_\alpha^{\text{COPY}} \leq d_\alpha^{\text{SPLIT}} \\ \mathbb{E}(O_\alpha^{\text{COPY}}) \geq \mathbb{E}(O_\alpha^{\text{SPLIT}}) \end{cases}$$

To prove the previous relations, it is enough to prove that $b_{A_1}, d_{A_1}, \mathbb{E}(O_{A_1})$ associated to the split node A_1 from the original node A in **Split** satisfy the following relations:

$$\begin{cases} b_{A_1} = 1 - \sqrt{1 - b_A} \leq b_A \\ d_{A_1} = \sqrt{d_A} \geq d_A \\ \mathbb{E}(O_{A_1}) = b_{A_1} + u_{A_1} a_{A_1} \\ = 1 - \sqrt{1 - b_A} + [\sqrt{d_A} + u_A - \sqrt{d_A}] \left[\frac{\sqrt{1 - b_A} - \sqrt{1 - b_A - a_A u_A}}{\sqrt{d_A} + u_A - \sqrt{d_A}} \right] \\ = 1 - \sqrt{1 - b_A - a_A u_A} \leq \mathbb{E}(O_A) = b_A + a_A u_A \end{cases}$$

1. Comparing b_α in Copy and Split:

Lemma A.35. $1 - \sqrt{1 - b_A} \leq b_A$

Proof.

$$0 \leq 1 - b_A \leq 1 \Rightarrow$$

$$\sqrt{1 - b_A} \geq 1 - b_A \Rightarrow$$

$$1 - \sqrt{1 - b_A} \leq b_A \Rightarrow$$

$$b_\alpha^{\text{SPLIT}} \leq b_\alpha^{\text{COPY}}$$

□

2. Comparing d_α in Copy and Split:

Lemma A.36. $\sqrt{d_A} \geq d_A$

Proof.

$$0 \leq d_A \leq 1 \Rightarrow$$

$$\sqrt{d_A} \geq d_A \Rightarrow$$

$$d_\alpha^{\text{SPLIT}} \geq d_\alpha^{\text{COPY}}$$

□

3. Comparing $\mathbb{E}(O_\alpha)$ in Copy and Split:

Lemma A.37. $1 - \sqrt{1 - b_A - a_A u_A} \leq b_A + a_A u_A$

Proof.

$$0 \leq b_A + a_A u_A \leq 1 \Rightarrow$$

$$\sqrt{1 - b_A - a_A u_A} \geq 1 - b_A - a_A u_A \Rightarrow$$

$$1 - \sqrt{1 - b_A - a_A u_A} \leq b_A + a_A u_A \Rightarrow$$

$$\mathbb{E}_{O_\alpha}^{\text{SPLIT}} \leq \mathbb{E}_{O_\alpha}^{\text{COPY}}$$

□

A.2 EXPERIMENTAL EVALUATION

In this section, we present some extra experiments made on several simple graphs having different topologies (*cf.* Table A.1). Besides the topology, these graphs vary in the percentage of common nodes and dependent paths they have.

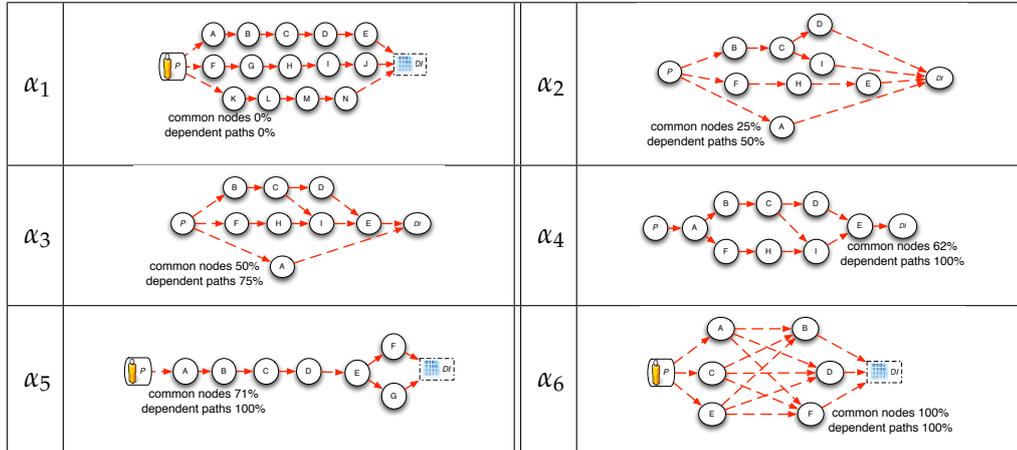


Table A.1 – Graphs that have different topologies.

We made the same experiment explained in Section 6.3.1. Table A.2 shows the obtained results.

As we mention in Section 6.3.1, when the graph has only independent paths, the three methods give the same exact results. Whereas, when the graph has dependent paths, some differences between the proposed methods appear. This difference become larger when the rate of dependent paths or common nodes increases.

Over a second step, we apply the same experiment presented in Section 6.3.2 on the graphs of Table A.1. Table A.3 shows the results of comparing system's opinion probability expectation $T_{\text{MTNA}}, T_{\text{COPY}}, T_{\text{SPLIT}}$ when $u = 0$ and the trust value T_{ST} resulting of using probability theory.

As we have already concluded in Section 6.3.2, in the graph that has independent paths α_1 , all results in **Copy**, **Split**, **mTNA** and **SocioTrust** are equal as expected. **Copy** shows the most accurate results for all the graphs. The average of **Split** is the worst but it shows a behavior more stable than **mTNA** because the standard deviation in **Split** is less than **mTNA**.

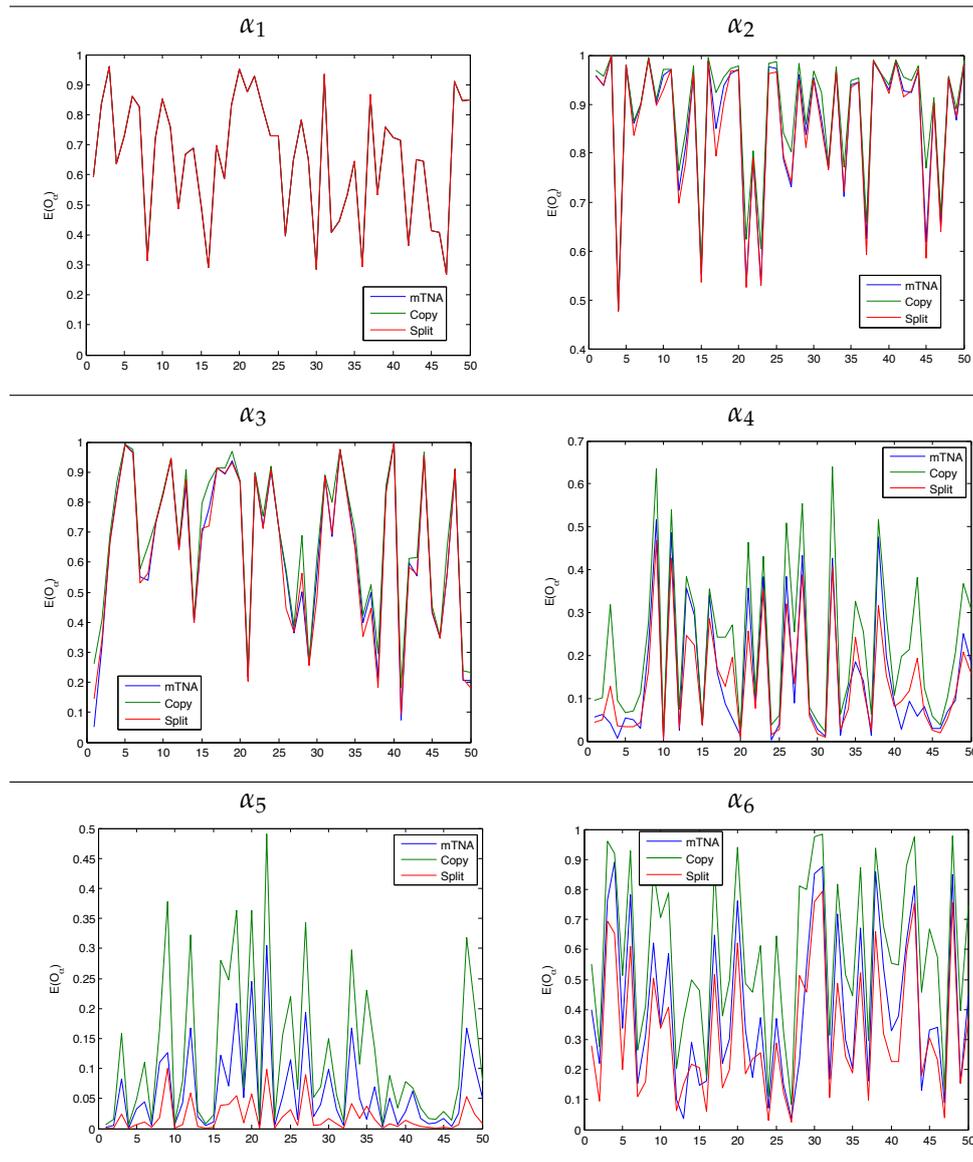


Table A.2 – Different graphs and their values of the probability expectation for 50 persons using the three methods *Copy*, *Split* and *mTNA*.

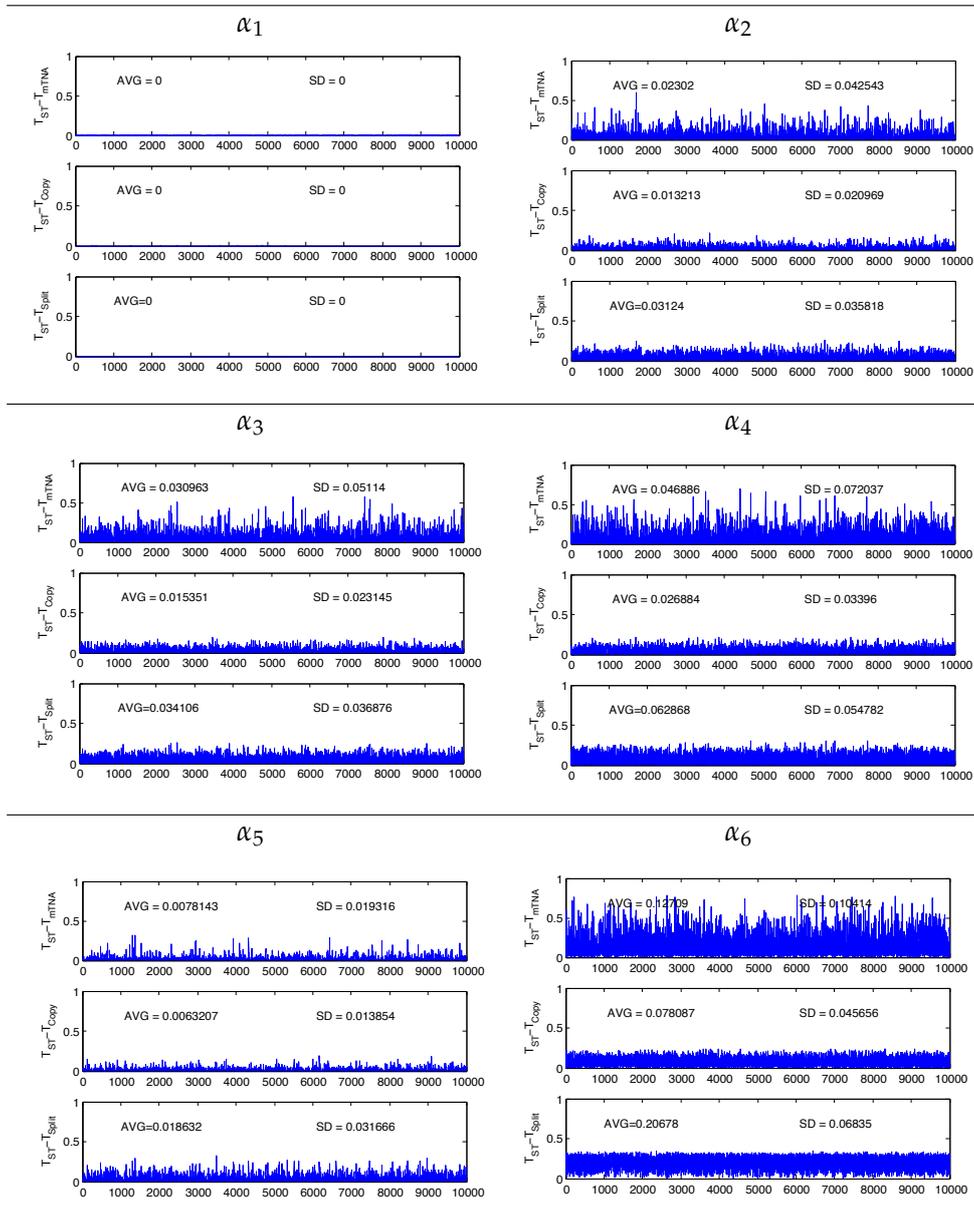


Table A.3 – The difference between the opinion’s probability expectation of a graph using SUBJECTIVE TRUST and the trust value resulting of using SOCIO TRUST.

A.3 PROPOSED SURVEY

Opinion on a node can be computed from user's negative or positive observations denoted r, s as proposed in [Jøso1] by the following relations.

$$\left\{ \begin{array}{l} b_x = \frac{r}{r+s+2} \\ d_x = \frac{s}{r+s+2} \\ u_x = \frac{2}{r+s+2} \end{array} \right\} \iff \left\{ \begin{array}{l} r = \frac{2b_x}{u_x} \\ s = \frac{2d_x}{u_x} \\ 1 = b_x + d_x + u_x \end{array} \right. \quad (\text{A.2})$$

Since our work focuses on local trust, the considered observations are the one made locally by a person on each node. This survey allows to collect the users' observations in order to build an opinion on a node. The proposed questions in this survey collect information about the user's usage of a node to estimate the uncertainty u . The negative observations of using a node is also demanded to estimate the value of d for a node. The value of b is computed by the relation $b = 1 - d - u$.

A node in the graph represents an artifact which is controlled by persons and is supported by physical resources. In the following, we present the proposed survey that let us build an opinion on each node.

1. Have you ever used Node N ?

- Yes
- No

If the answer is "No", then uncertainty is complete and the user's opinion on this node is set as $(0, 0, 1, 0.5)$.

If the answer of Question 1 is "Yes", the following is demanded:

2. At which frequency do you use Node N ?

- Everyday
- From time to time

If the answer is "Everyday", then the uncertainty should be null as the user has an intensive usage of the node ($u = 0$).

If the answer of Question 2 is "From time to time", the following is demanded:

3. More precisely, how many times have you used Node N ?

- 1 time (from Relations A.2 $\Rightarrow u = 2/(1+2) = 0.67$).
- 2 times (from Relations A.2 $\Rightarrow u = 2/(2+2) = 0.5$).
- 3 times (from Relations A.2 $\Rightarrow u = 2/(3+2) = 0.4$).
- Around 5 times (from Relations A.2 $\Rightarrow u \approx 2/(5+2) \approx 0.28$).
- Around 10 times (from Relations A.2 $\Rightarrow u \approx 2/(10+2) \approx 0.17$).
- Around 20 times (from Relations A.2 $\Rightarrow u \approx 2/(20+2) \approx 0.09$).
- Around 50 times (from Relations A.2 $\Rightarrow u \approx 2/(50+2) \approx 0.04$).
- Around 100 times (from Relations A.2 $\Rightarrow u \approx 2/(100+2) \approx 0.02$).

- More than 100 times (from Relations A.2 $\Rightarrow u \approx 0$).

The answer of this questions allow to evaluate the value of u from the relation $u = 2 / (\text{total observations} + 2)$, introduced in subjective logic.

These last 3 question let us to determine the value of u . It remain to evaluate values of d or b . Since users remember their negative observations more easily than their positive ones, the following questions are asked.

If the answer of Question 1 is "Yes", the following is demanded:

4. Have you ever had a problem with the Node N ?

- Yes
- No

The answer "No" infers a null disbelief and $d = 0$ since the user has never got negative observations.

If the answer of Question 4 is "Yes" and the answer of Question 2 is "Everyday", the following is demanded:

5. How much do you estimate that you had problems with this node comparing to the total times you use this node (the answer should be given as a percentage)?

The given value correspond to the value of d .

If the answer of Question 4 is "Yes" and the answer of Question 2 is "From time to time", the following is demanded:

6. How many times you had a problem with the node N ?

d can be computed from the Relation A.2 $\Rightarrow d = \frac{s}{r+s+2} = \frac{\text{Answer 6}}{((\text{Answer 3})+2)}$

At this stage, we have the value of d and u . Thus, the value of b can be computed from the aforementioned relation $b = 1 - d - u$.

We modeled a subpart of the LINA research laboratory¹ system using SOCIOPATH. We applied the rules of SOCIOPATH on this system for the activity "a user accesses a document `todo` that is stored on the SVN server of LINA" (cf. Section 5.4.3). Figure 5.6 presents the DAG for this activity.

20 members of LINA participated and answered these questions for each node. The value of a is equal to 0.5 for each node since it is a prior probability in the absence of the evidence. For instance, P_1 answers the questions about G as following:

- Question 1: Yes.
- Question 2: From time to time.
- Question 3: Around 10 times $\Rightarrow u = 0.17$.
- Question 4: No $\Rightarrow d = 0 \Rightarrow b = 0.83$.

Thus, $w_G = (0.83, 0, 0.17, 0.5)$ for P_1 .

Another example is the opinion of P_{10} about G . P_{10} answers the questions about G as following:

¹<https://www.lina.univ-nantes.fr/>

- Question 1: Yes.
- Question 2: Everyday $\Rightarrow u = 0$.
- Question 4: Yes.
- Question 5: 20% $\Rightarrow d = 0.2 \Rightarrow b = 0.8$.

Thus, $w_G = (0.8, 0.2, 0, 0.5)$ for P_{10} .

The full result of this survey is given in Table 6.1.

Titre Comblent le fossé entre les mondes sociaux et numériques : modélisation de systèmes et évaluation de la confiance.

Résumé Aujourd'hui, les systèmes numériques sont interconnectés au travers d'architectures complexes. Les participants à ces systèmes réalisent activités telles que communiquer ou partager des données. Humains, matériels et logiciels sont impliqués dans ces activités, de sorte qu'un système peut être décrit comme la représentation de deux mondes, le monde social et le monde numérique et leurs relations. L'évaluation de ces systèmes est limitée aux aspects techniques. Aujourd'hui, la confiance envers les est devenue un facteur important dans les procédures d'évaluation. Dans ce contexte, nous soulevons deux questions : comment formaliser les entités composant un système ainsi que leurs relations pour une activité particulière ? et comment évaluer la confiance envers un système pour cette activité ? Nos contributions sont divisées en deux parties. La première partie propose un métamodèle formel, nommé SocioPath, qui permet de modéliser un système avec les entités du monde social et numérique et leurs relations. La deuxième partie consiste à évaluer la confiance des utilisateurs envers les systèmes qu'ils utilisent pour une activité donnée. Nous proposons une approche, nommé SocioTrust permettant de calculer le niveau de confiance, qui utilise la théorie des probabilités. Puis nous proposons une seconde approche, sous le nom de SubjectiveTrust, pour prendre en compte l'incertitude dans les valeurs de confiance, cette approche se basant sur la logique subjective.

Mots-clés Modélisation, Dépendance, Confiance, Logique subjective.

Title Bridging the Gap between Social and Digital Worlds: System Modeling and Trust Evaluation.

Abstract Nowadays, digital systems are connected through complex architectures. Participants to these systems perform activities like chatting or sharing data. Persons, physical and digital resources are involved in these activities, such that a system can be considered as a representation of two worlds, the social world and the digital world and their relations. Evaluating these systems is generally limited to technical aspects. Today, trust becomes an important key in the evaluation process. In this context, we raise two questions: how to formalize the entities that compose a system and their relations for a particular activity? and how to evaluate trust in a system for this activity? Our contributions are divided into two parts. The first part proposes a formal metamodel named SocioPath, to model a system with all entities of social and digital worlds and their relations. The second part evaluates the users' trust in the systems they use for a given activity. We propose an approach named, SocioTrust, to compute the user's trust in a system using probability theory. Then we propose a second approach named, SubjectiveTrust that takes into account the uncertainty in the trust values. This approach is based on subjective logic.

Keywords Modeling, Dependencies, Trust, Subjective logic.