Knowledge Provenance: An Approach to Modeling and Maintaining The Evolution and Validity of Knowledge

by

Jingwei Huang

A thesis submitted in conformity with the requirements
for the degree of Ph.D.
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

Abstract

Knowledge Provenance: An Approach to Modeling and Maintaining The Evolution and Validity of Knowledge

Jingwei Huang

Ph.D.

Graduate Department of Mechanical and Industrial Engineering

University of Toronto

2008

The Web has become an open decentralized global information / knowledge repository, a platform for distributed computing and global electronic markets, where people are confronted with information of unknown sources, and need to interact with "strangers". This makes trust and the validity of information in cyberspace arise as crucial issues.

This thesis proposes knowledge provenance (KP) as a formal approach to determining the origin and validity of information / knowledge on the Web, by means of modeling and maintaining the information sources, information dependencies, and trust structures. We conceptualize and axiomatize KP ontology including static KP and dynamic KP. The proposed KP ontology, provides a formal representation of linking *trust* in information creators and *belief* in the information created; lays a foundation for further study of knowledge provenance; provides logical systems for provenance reasoning by machines. The web ontology of KP can be used to annotate web information; and KP reasoner can be used as a tool to trace the origin and to determine the validity of Web information.

Since knowledge provenance is based on trust in information sources, this thesis also proposes a logical theory of trust in epistemic logic and situation calculus. In particular, we formally define the semantics of trust; from it, we identify two types of trust: *trust in belief* and *trust in performance*; reveal and prove that *trust in belief* is transitive; *trust in performance* is not, but by *trust in belief*, *trust in performance* can propagate in

social networks; by using situation calculus in trust formalization, the context of trust is formally represented by reified fluents; we also propose a distributed logical model for trust reasoning using social networks, by which each agent's private data about trust relationships can be protected. This study provides a formal theoretical analysis on the transitivity of trust, which supports trust propagation in social networks. This study of trust supports not only knowledge provenance but also the general trust modeling in cyberspace.

Acknowledgements

First of all, I would like to thank my supervisor Professor Mark S. Fox, for his support and guidance. His insight, methodology and criticism, are invaluable not only in this work, but also in my future research. Mark's sound of correcting my pronunciation still vibrates in my mind.

I wish to thank my committee members Professor Michael Gruninger and Professor Thodoros Topaloglou. Michael's mentoring on logics helps me a lot to improve the quality of this thesis. I also very appreciate Thodoros' comments and many kindly helps on this thesis, general research thinking and writing.

I also wish to give sincere thanks to my external examiner Professor Victor Lesser. His appraisal and questions inspire me to think deeper and wider on both this work and research methodology.

Thanks to Professor Eric Yu. His valuable feedbacks and questions help me to improve this research.

During my time at Toronto, I got helps from many people. Sincere thanks to Dr. Harold Boley for his important comments. I also appreciate the feedbacks from Professor Mark Chignell and Professor Joseph Paradi. Thanks to Dr. Zhongdong Zhang, Dr. Mihai Barbuceanu, and Dr. Yannick Lallement, for the discussions with them. I very appreciate Zhongdong's encouragement in my difficult time and his many helpful tips in software development. Thanks to Professor Jun Wu for her helps and support. Thanks to Professor Michael Carter and Professor Chris Beck for their kindly helps.

Thanks to the English Language & Writing Support program offered by the School of Graduate Studies in the university. I largely benefit from this program on surviving in English academic world. Particularly, I wish to thank Dr. Valia Spiliotopoulos for her many kindly helps on improving my English.

Finally, thanks to my parents, Yujun Huang and Yunhua Chen, and my wife, Mingming Wang, for their understanding and constant support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis proposes *knowledge provenance* (hereafter referred to as *KP*) to address the problem of how to determine the origin and validity of information /knowledge on the Web by modeling and maintaining information sources, information dependencies and trust structures. In the following sections, we introduce the background and motivations of the research, the outline of this thesis and the major contributions.

1.1 Background and Motivations

This thesis is motivated by two tightly related problems. The first problem is how to determine the validity of the information / knowledge on the web. Because of the widespread use of the Web and the development of Web technologies and telecommunication technologies, the Web has fast become an open decentralized global information/knowledge repository, where anyone is able to produce and disseminate information, so that the information on the Web may be true or false, current or outdated; however, few tools exist to discern the difference. Information validity has become a serious problem on the Web. For example, in 1999, two individuals posted fraudulent corporate information on electronic bulletin boards, which caused the stock price of a company (NEI) to soar from $0.13 to $15, resulting in their making a profit of more than

$350,000 [125]. In order to solve this problem, methods and tools need to be developed to determine the validity of web information. Based on traditional information quality evaluation criteria, researchers developed web information quality evaluation criteria such as *authority*, *accuracy*, *objectivity*, *currency* and *coverage* [9] [3]. Nevertheless, most proposed information quality evaluation models are not formal models. In other words, by using these models, human users need to be involved in information quality evaluation process. So that these models cannot be automated to judge the quality of information. This thesis aims to develop a formal model used to determine the origin and validity of web information by computers. The origin of a piece of information can help information users to determine the validity of the information; the origin is also an important part of context of this information, which can help information users in using this information.

The second problem to be addressed in this thesis is trust problem on the web. Due to the development of new Web technologies such as the Semantic Web[14], web services[143] and P2P[132], the Web is not only a global information/knowledge repository, but also a distributed computing platform. In this new territory of cyberspace, people, organizations and software agents need to interact with "strangers" i.e. entities with unknown identifications. In such an open and uncertain territory, can people trust "strangers"?

Interest in addressing this issue has appeared under the umbrella of the "*Web of Trust*" which is identified as the top layer of the Semantic Web and is still in its infant stage of development ([12] slides 12; [11] slides 26&27). *Web of Trust* aims to create a trusted Web. Basically trust is established in the interaction between two entities. However, any single entity only has a finite number of direct trust relationships, which cannot meet the needs of various interaction with unknown or unfamiliar entities on the Web. As a promising remedy to this problem, social networks-based trust, in which A trusts B, B trusts C, so A indirectly trusts C, is receiving considerable attention. A necessary condition for trust propagation in social networks is that trust needs to be transitive. However, is trust transitive? What types of trust are transitive and why?

There are neither theories nor models found so far to answer these questions in a formal manner. Most models either directly assume trust transitive or do not give a formal discussion of why trust is transitive. To fill this gap, this thesis will build a logical theory of trust that formally defines the semantics of trust and derive the transitivity of trust and the conditions for trust propagation in social networks.

The first problem of Web information validity may be reduced to a specific trust problem: to determine the validity of information by evaluating the trustworthiness of information sources. However, the drawbacks of this pure trust-based approach are: the validity of information is determined only by the trust placed in information sources; other features of information such as information dependencies are neglected. This thesis proposes KP to determine the validity of information by taking both trust and information dependencies into account.

1.2 Thesis Outline

This thesis is aimed at creating a logical theory of knowledge provenance in the form of ontology which can be used to determine the validity and origins of the information/knowledge on the Web. The basic questions KP attempts to answer include:

- Can this information be believed to be true?
- Who created it?
- Can its creator be trusted?
- What does it depend on?
- Can the information it depends on be believed to be true?
- Is trust transitive?
- What types of trust are transitive and why?

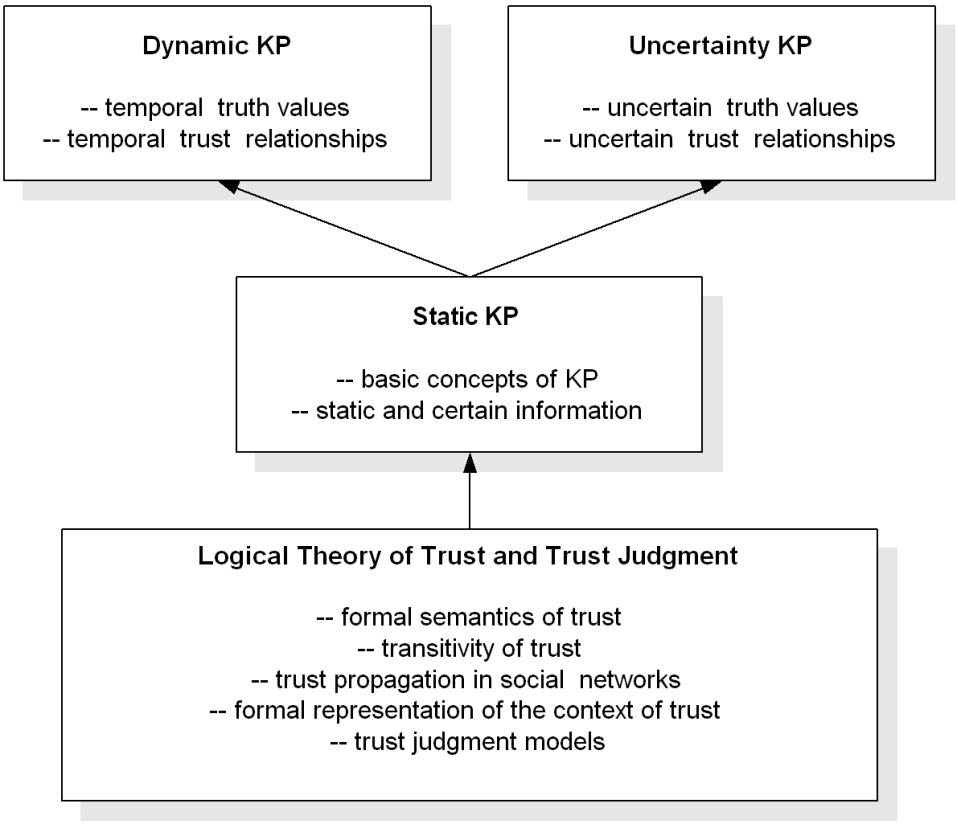

Figure 1.1: The Structure of KP models

- Is trust able to propagate through social networks?

By using technologies developed in the semantic web [11], and based on the logical theory of KP, we design a web ontology of KP used to annotate web documents with KP metadata to describe the provenance-related attributes, such as who is the proposition creator and what is the support proposition which this proposition depends on; develop a web ontology of trust used to define personalized trust relationships in social networks; develop a KP reasoner used to trace KP metadata in web documents across web pages, combining information sources and dependencies, as well as trust relationships, to deduce the origin and validity of tagged information.

The focus of this thesis is a logical theory of KP. We identify four modules in KP as shown in figure 1.1. From simple to complex, we start from static KP to develop the basic concepts of KP in the simplest case in which only deterministic and static information

is considered; following static KP, we extend it into dynamic KP; then, we turn to trust modeling which supports KP. In this thesis, we do not cover uncertain KP. Uncertain KP is studied in [87] and [93]. Regarding the relation between KP and trust, on one hand, KP concerns information trustworthiness – a specific trust problem; on the other hand, KP uses a trust judgment model to determine whether the information sources cab be trusted.

In Chapter 2, the research relevant to information validity judgment is reviewed in three aspects: the information characteristic based judgments, provenance based judgments and trust based judgments.

In Chapter 3, static KP, which focuses on certain and static information, is studied. On one hand, static KP can be used in the situation where the validity of information is either true or false and the validity does not change over time; on the other hand, static KP provides fundamental concepts for constructing more general KP models.

In Chapter 4, dynamic KP, which addresses the provenance problems where the validity of information and trust relationships change over time, is studied.

In Chapter 5, a logical theory of trust in the form of ontology is constructed. In the theory, the semantics of trust is formally defined in epistemic logic and situation calculus; from the formal semantics, the transitivity of *trust in belief* and the condition for *trust in performance* propagation in social networks are revealed and proved; based on this trust ontology, trust networks, a straightforward representation of trust reasoning using social networks, is constructed.

In chapter 6, a social network-based distributed trust reasoning model is proposed, and constructed in situation calculus. This distributed model can be implemented with web services, then each entity in social networks only answers whether to trust a questioned entity, and need not to publish personal trust data online, so that this model facilitates social network-based trust reasoning, and at the same time privacy is better protected.

In Chapter 7, an application of KP in financial information is developed.

Finally, in Chapter 8, a summary of the contributions of this thesis and a discussion about future work is given.

1.3 Major Contributions

The major contributions of this thesis are listed as follows.

1. This thesis proposes the concepts of knowledge provenance, and constructs a logical theory of knowledge provenance in the form of ontology for determining the origin and validity of information / knowledge on the Web. In particular, this work includes:

    - axiomatization of a static KP ontology that defines the basics of KP, and is used for determining the validity of static and certain information;
    - axiomatization of a dynamic KP ontology used for determining the validity of information whose validity changes over time;
    - a web ontology of KP in OWL used for annotating web information;
    - an application case study to demonstrate how to combine KP and XBRL for financial information provenance.

2. This thesis constructs a logical theory of trust, by formalizing trust in epistemic logic and situation calculus, including:

    - formalization of the semantics of trust based upon belief, which gives a formal explicit definition of trust, so that it facilitates the use of trust in formal systems;
    - identification of two types of trust: *trust in belief* and *trust in performance*;

- revealing and proof of that trust in belief is transitive, trust in performance is not, but can propagate in social networks by trust in belief;
- a trust networks model, which is a straightforward graph representation of the computationally intensive logic model of trust, and turns the logical reasoning of trust into trust path searching, so that it is easier to be used in practice;
- a distributed trust reasoning model, which facilitates social network-based trust judgment, and at the same time better protects the privacy of entities in social networks regarding data on trust relationships.

Chapter 2

Literature Review

In this chapter, we present a review of the research related to knowledge provenance. As discussed in Chapter 1, the problem to be solved in KP is how to determine the validity of information/knowledge on the Web. In accordance with judgment approaches, we divide the related research into 3 classes: (1)judgments based on the characteristics of the information; (2) judgments based on trust; (3)judgments based on the provenance of the information. These three categories of research are examined in sections 1, 2, and 3 separately. Furthermore, section 4 gives a review of the related technologies that support KP such as the semantic web, digital signature, and situation calculus. Finally, we give a summary of this literature review in section 5.

2.1 Information Characteristic Based Judgment

The validity of a piece of information may be judged by analyzing the characteristics of the information. We examine the related research conducted in management information systems community and library and information science community separately.

2.1.1 Information Quality Assessment

Research on information quality assessment is originally developed from data quality assessment for databases [183]. MIT Total Data Quality Management program (TDQM) developed AIMQ method [111] to improve information quality in organizations. AIMQ includes 3 components. (1) IQ dimension, (2) Questionnaire methods to assess IQ, and (3) IQ Analysis, e.g. using benchmark to find gap to best practice, and finding opinion gap between different classes of reviewers.

TDQM defines information quality in 16 dimensions organized in 4 categories: intrinsic, contextual, representational and accessibility, reflecting the properties of information itself, relevance to user's task, information form, and accessibility respectively [111]. The proposed criteria [148] relevant to our research are as follows (Note: in their context, data is used as a general term that includes the meaning of information). (1) Believability, the extent to which data is regarded as true and credible; (2) Free-of-Error, the extent to which data is correct and reliable; (3) Objective, the extent to which data is unbiased, unprejudiced, and impartial; (4) Reputation, the extent to which data is highly regarded in term of its source or content. (5) Timeliness, the extent to which the data is sufficiently up to date for the task at hand.

Naumann and Rolker [138] unify IQ criteria from different sources, and give a list of synonymous criteria and explanation to them. Regarding information quality assessment methods, since "IQ criteria are often of subjective nature and can therefore not be assessed automatically" [138], usually, questionnaire method is applied for IQ assessment, and then functors such as *min*, *max*, or *weighted average* are applied to get overall evaluations.

TDQM explored an approach that derives an overall data quality value from detailed criteria or indicators by using local dominance relationships among quality parameters [184]. Local dominance relationships can be illustrated by the following example: "timeliness is more important than the credibility of a source for this data, except when

timeliness is low".

This research area mainly focuses on Information Quality in Management Information Systems. The IQ assessment is human-dominated process, and the information sources to be assessed are determinate.

2.1.2 Web Information Quality Evaluation

In library and information science, information quality (IQ) evaluation criteria have been studied for IQ management and relevance judgments. Relevance judgments are information users' judgments regarding whether the information found in a search meets the users' needs. In relevance judgments, except topic relevance, information quality is another necessary criterion. Validity is one of the most important aspects of information quality.

A set of information quality evaluation criteria, which are used by scholars in academic environment who perform information retrieval from traditional text-based print documents, has been identified in an empirical study [8] for the purpose of relevance judgments. Another empirical study [159] examined the criteria used by professionals who perform weather-related information retrieval from various sources such as weather documents, information systems, the mass media, and interpersonal communications. A comparison of these two studies found that the general criteria employed in two cases are the same or similar, and this result suggests that "the existence of a finite range of criteria that are applied across types of users, information problem situations, and information sources" [9].

The criteria common to both studies and relevant to validity are listed as follows.

- **Depth/Scope/Specificity.** The extent to which information has sufficient detail and range to meet user's needs.

- **Accuracy/validity.** "The extent to which information is accurate, correct, or

valid."

- **Clarity.** "The extent to which information is presented in a clear and well-organized manner."
- **Currency.** "The extent to which information is current, recent, timely, or up-to-date."
- **Tangibility.** The extent to which "information relates to real, tangible issue", or "definite, proven information is given."
- **Quality of Sources.** The extent to which "source is reputable, trusted, expert", or "general standards or specific qualities can be assumed based on the source".
- **Verification.** "The extent to which information is consistent with, or supported by other information in the field."

Although these criteria are proposed to evaluate traditional information, they are basically universal for all kinds of media including Internet information.

Many criteria used in traditional information evaluation have been recommended by researchers and libraries (see [3], [171]; [174]; [31]) to evaluate web information content. Most of these criteria can be covered by five general criteria: **Authority**, **Accuracy**, **Objectivity**, **Currency**, and **Coverage**. The detailed criteria can be found in [3], [154] and [171]. Obviously, Barry and Schamber's criteria discussed earlier in this section are very close to these 5 general criteria.

From the perspective of information science, these studies discussed above provide us with useful clues regarding what features of information should be considered in constructing a KP model. However, the information quality evaluation methods in these studies are informal. That is to say, the evaluations is supposed to be conducted by information users in an informal manner, typically in a question-answer style. Information users need to answer each evaluation question and to make overall evaluations by

themselves. There are no formal models to reveal the relations among these criteria, and there are no automated tools to support such analysis.

The difficulties in constructing formal models are: (1) many criteria are content-dependent; and (2) many criteria "are of subjective nature and can therefore not be assessed automatically" [138]. There are two approaches to solving the problems. One is semantic annotation of Web information. We will discuss this stream of research in section 4. Another approach is the use of content-independent criteria.

The criteria regarding authority (or quality of sources) are content-independent; furthermore, authority is highly associated with information quality. Some empirical studies (e.g. [154]) show that the source credibility and authority plays an important role in information quality evaluation, and "people depend upon such judgments of source authority and credibility more in the Web environment than in the print environment." The connection between authority and information quality is also supported in theory by Wilson's book *Second-Hand Knowledge* (1983)[186]. In the book, the author developed a theory of "cognitive authority". "Cognitive authority", different from "administrative authority", is addressed as "influence on one's thought that one would consciously recognize as proper". Note that the word "authority" we used earlier in this section is referred to "cognitive authority". According to the author, the cognitive authority of a text may be tested from four characteristics: (1) the author; (2) the publisher; (3) the document type such as dictionaries; (4) the intrinsic plausibility of the text. Especially, relevant to the connection between authority and information quality, the author argued that "a person's authority can be transferred to his work as long as the work falls within the sphere of his authority". This theory is tightly related to trust based judgment which we will discuss in the next section.

2.2 Trust and Trust Based Judgment

As addressed by Wilson (1983), "we can trust a text if it is the work of an individual or group of individuals whom we can trust". Trust is an important and efficient approach to judge the validity of information / knowledge. This section discuss the related research on trust and trust modeling.

Trust is widely concerned by many disciplines such as psychology, philosophy, sociology, politics, economics, management sciences, and computer science. In the following, we discuss (1) trust conceptualization mainly in social sciences; then we turn to the trust formalization mainly in computer science and management sciences including (2) trust management developed in computer network security; (3) trust in distributed AI; (4) trust in social networks; and (5) trust in e-business.

2.2.1 Trust Conceptualization

Trust is a complex social phenomenon. In order to construct a formal model of trust on the web, it is important to learn the concepts, structure and nature of trust from the theories of trust developed in social sciences. In the following, we discuss the major views of trust.

Trust as Decision

Since 1950s, Deutsch [38, 40] studied trust in the context of cooperation . He defined trust as a trusting choice under an uncertain situation: (1) this choice may lead to two possible outcomes: a beneficial one and a harmful one; (2) which outcome occurs depends on the behavior of another individual; (3) the strength of the harmful outcome is stronger than the beneficial one. A set of conditions and hypotheses on the decision of trust were given. The hypothesis to make trusting choice is that the difference of the expected utility of beneficial outcome and the expected utility of harmful outcome is greater than

the decision maker's personal "security level". However, how to calculate this security level remains unclear. Coleman [29] proposed a condition to place trust purely based on the postulate of maximization of utility. Coleman's condition is similar to Deutsch's trusting choice hypothesis but without that "security level". A common problem of these two conditions is that the utility of choosing *not trust* is missed in decision. A decision should be made based on the expected utilities of both *trusting* and *not trusting*.

The definition of trust in Deutsch's approach is to regard trust as decision in the cooperation/ interaction between individuals. Deutsch's work lays an important part of foundations for the formalization of trust. However, this view of trust does not reveal the conceptual structure of trust.

Trust as Psychological State

A large body of research has contributed to the evolution of the conceptualization of trust. Rotter [155] defined "interpersonal trust" as "an expectancy held by an individual or a group that the word, promise, verbal or written statement of another individual or group can be relied on." This definition reveals the essential aspect of trust – the expectancy that the trusted party will keep its word.

Many researchers recognized that trust is associated with risk. For example, Gambetta [57] addressed that trust is fragile due to the limit of knowledge, foresight and the uncertainty of the behaviors of the trusted agent(s). Mayer et al [124] further incorporated risk factor into the definition of trust. The authors defined trust as "the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party." This definition is one of most frequently cited ones.

Rousseau et al [156] synthesized the concepts of trust in different disciplines and addressed that "Trust, as the willingness to be vulnerable under condition of risk and

interdependence, is a psychological state". The authors further emphasize that trust is not a behavior or a choice but an underlying psychological condition of these actions.

There are many other views of trust, for example, from the view of economists, trust is "implicit contracting" [196]; in Fukuyama's view, "trust is the expectation that arises within a community of regular, honest, and cooperative behavior, based on commonly shared norms"[56].

Finally, Blomqvist [19] analyzed the concepts of trust in different fields and compared with trust-related concepts such as incredibility, confidence and so forth. The author presented a wide landscape of trust studies in social sciences. McKnight and Chervany [128] examined 65 definitions in different disciplines and proposed a topology of trust, which is helpful to distinguish and to connect different types of trust.

Types of Trust

Trust can be classified into different types. In accordance with the psychological states of making trust choice in different decision situations, Deustch [40] presented many different types of trust. Typical examples are "trust as confidence", "trust as innocence", "trust as virtue" and "trust as gambling". In the dimensions of rationality and emotionality, Lewis and Weigert [113] presented a spectrum of trust. For examples, "cognitive trust" is the trust with high rationality and low emotionality; "emotional trust" is the trust with low rationality but high emotionality; in extreme situations, high rationality but no emotionality corresponds to "rational predication"; and no rationality but high emotionality corresponds to "faith".

More relevant to our work, trust can also be classified by the sources of trust. Some scholars may refer the term of "sources of trust" to the characteristics of trustee, which make trustor trust, such as trustee's "competency" and "goodwill". By this term, we mean where or how trust comes from. The most direct and common source of trust is interaction experience. The trust with this type of source is called *interpersonal trust*, or

more generally, *inter-individual trust*. This type of trust is the most fundamental type of trust, so that it is the major body of trust study. However, a considerable attention has given to "*system trust*" - a different mechanism of trust emerging at the turn of the last century to meet people's increasing demands of more and more interaction with strangers in industrial society (a bigger world ever before). This situation is very similar to the trust problem we are facing today in the cyberspace.

System trust is first revealed by Luhmann in his influential book *Trust and Power* [118]. *System trust* is the trust placed in the function of a system in society. System trust has many manifestations. Barber [7] studied three types of expectations: (1) expectation of "persistence and fulfillment of the natural and the moral social orders"; (2) expectation of "technically competent role performance" of professionals; (3) expectation of the "fiduciary obligations and responsibilities" of the partners in interaction. Zucker [196] examined the evolution of the trust mechanism in American economic system and identified three modes of "trust production" (trust building): (1) "process-based", in which trust is built on past history of interaction; (2) "characteristic-based", in which trust is dependent on "social similarity", such as ethnicity, age and gender; (3) "institutional-based", in which trust is built on "formal social structure" comprising of "membership in a subculture" and "intermediary mechanism", such as regulation, legislation, functions of governments and banks. The "process-based" actually is interpersonal trust, and the latter two (types (2) and (3)) are the manifestations of *system trust*. System trust is based on the predictable behaviors of trustee in a (social or natural) system.

Most recently, trust propagation in social networks is receiving much attention. This type of trust is based on the transitivity of trust. We call this type of trust as "relational trust" and will discuss its related research in subsection "trust in social networks".

2.2.2 Trust Management

In computer science, trust was initially a concern of the security community. For the purposes of secure web access control, Blaze et al [17] first proposed "decentralized trust management" to separate trust management from applications. The authors introduced the fundamental concepts of policy, credential and trust relationship, and they also developed the PolicyMaker system. Chu [28] introduced trust protocol in REFEREE system; Yahalom et al [191] introduced recommendation type of trust in security system; Khare and Rifkin [102] proposed basic principles of trust management. However, Trust Management focuses on one specific type of trust – "is this individual trusted to access or to do a specific operation in my system?" The concerns from "Web of Trust" are more general, e.g., whether the information created or action conducted by an individual could be trusted. Recently, research interests in trust management field are expanding from security to general purposes, especially, towards trust management in e-commerce.

2.2.3 Trust in Distributed AI

Trust is also a concern of the distributed AI community.

Based on a thorough examination of the trust concepts developed in social sciences, Marsh [122] constructed a quantitative model of trust as a formal tool to study trust decisions in the interaction among agents by means of quantitatively simulating the psychological process for those decisions. In this model, trust is associated with situations and the utilities of situations. This work is among the first to formalize trust. However, several limitations exist. For example, this study is limited to only interpersonal trust; although "situation" is introduced in the trust model, the model does not clearly distinguish two types of "situations": (1) the situation in which the trustee conducts the expected things; (2) the situation in which trustor makes a trust decision.

Demolombe [37] constructed a modal logic model of trust. Based on operators *belief*

and *strong belief*, the author defined the formal semantics of the trusts in trustees' properties of sincerity, credibility, cooperativity and vigilance in delivering information. This work is among the first to formalize trust using *belief*. However, this approach suffers from the problem regarding what is the possible world semantics of belief (see discussion in 2.4.3). The author seems to treat belief as "necessitation", which leads to that believed proposition must be true, but in fact belief is not necessarily true. Furthermore, similar to the "logical omniscience" problems in epistemic logic, trust regarding to vigilance (in which trustor a strongly believes that if a proposition is true then trustee b believes this proposition) is not true in the real world. In addition, the logic used in this model seems beyond first order modal logic. These features may make the model difficult to use. Another problem is that this model does not tell the difference between the information believed and the one created by the trustee, which have different trust properties. Finally, the trust addressed focuses on interpersonal trust, so that transitivity of trust is not studied.

Falcone & Castelfranchi [46] constructed a socio-cognitive model of trust that suggests to calculate the trustworthiness of an agent about a task in a context by counting the degree of ability and the degree of willingness (motivation) of the agent. It is interesting to connect trust with context. However, It is hard to measure the degree of willingness of the trustee.

Gans et al [59] developed a Trust-Confidence-Distrust model for representing collaboration in agent networks. Perhaps because trust problem in DAI arises in the interaction among agents, all the work discussed above focus on inter-individual trust.

Ramchurn et al [149] classified the approaches to trust in multi-agent systems into individual-level trust and system-level trust. Individual-level focuses on learning, trust evolving, trust evaluation; and system-level focuses on the mechanisms of trustworthy interaction and reputation building. They discussed the approach to reputation building and evaluation using social networks.

In addition, several technologies developed in autonomous agents and multi-agent systems are tightly related to trust formalization, for examples, belief, goals, intention, commitments, coordination and so forth [21, 187, 50, 193].

2.2.4 Trust in Social Networks

A "social network" is a network representing the relationships between individuals or organizations, indicating the ways in which they are connected through various social familiarities ranging from casual acquaintance to close familial bonds [185]. Milgram's experiments in 1960s [131] reveal an interesting finding – six degree of separation, which suggests that any two randomly chosen individuals in America are connected, on average, by a chain of 6 acquaintances. This finding provides a compelling evidence for "small world phenomenon" - a hypothesis that any two individuals in a social network are likely to be connected by a short chain of acquaintances [106]. Dodds et al [43] conducted an experiment in email users which further confirms the small world theory in cyberspace.

In recent years, trust models based on *social networks* are receiving considerable attention. Particularly, the trend is powered by "web of trust" which is identified as the top layer of the semantic web.

The concept of "web of trust" perhaps is first developed in PGP as a trust model used for public key validation by using social networks. However, "trust" in PGP is specifically on public key validation. FOAF project (http://foaf-project.org/) attempts to create social networks on the web by facilitating people to describe acquaintance relationships in machine-readable web pages. Although acquaintance relationships are not trust relationships, FOAF is a good starting point. Recently, many models of trust on the web using social networks have emerged. For examples, Yu and Singh [192] constructed a model of reputation (trust) updating and propagation using the testimonies from the trustee's neighbors; Golbeck et al [66] extended the acquaintance relationships in FOAF model by introducing levels of trust and applied the model for filtering emails; Richard-

son et al [153] proposed an algebra representation of trust propagation and applied it in bibliography recommendation; Guha and Kumar [79] constructed a trust propagation model considering distrust; Abdul-Rahman and Hailes [1] proposed a trust model including "direct trust" and "recommender trust", in which trust propagates in social networks; Josang et al [99] argued that trust is not always transitive and "referral trust" (the same as "recommender trust") is transitive. However, they did not reveal why recommendation is transitive in a formal manner.

A common perspective of most of these models is that trust propagates in social networks. However, is trust transitive? What types of trust are transitive and why? Few theories and models found have answered these questions in formal manner. The models found either directly assume trust transitive or do not give formal discussion why trust is transitive, due to no formal representation of the semantics of trust.

Most of social networks-based trust models are quantitative models in which a trust degree is interpreted as subjective probability. The advantages of quantitative models include: (1) the uncertainty feature of trust is addressed; (2) trust models may be built based on the well established uncertainty theories. However, on the other hand, many quantitative models still suffer from the lack of an explicit formal interpretation of the meaning of trust degree, due to the lack of formal semantics of trust, so that the trust degree is defined quite subjective. Another problem is that probabilistic models usually need many parameters, but it is difficult to obtain these parameters. Perhaps for this reason, in many quantitative models, on one hand, trust degree is interpreted as probability, on the other hand, the models used for computing trust degrees are not based on probability theory. It is a problem regarding how to explain those models in probability theory.

2.2.5 Trust in e-Commerce

Both the conceptual models and formal models of trust developed in social sciences and computer science are widely applied to analyze the trust in online trade [128] [108] [100]. In addition to trust modeling's manifestations and applications in the field of e-commerce, two interesting approaches developed: reputation systems and third party certification.

A "reputation system" collects, aggregates and distributes the feedback about online trade participants's past behaviors [151]. Typical examples include: eBay, amazon and epinions. The "reputation" of a participant is the aggregated evaluation from other participants regarding past interaction, which could be approximately regarded as the trust obtained from a community. However, "reputation systems" have several limitations [151]. One major problem is unfair rating [194]; another is the missing of contexts, for example, from the overall "reputation", users cannot find whether a retailer is only good at some specific products like books.

Coming from the concerns of security and privacy, "trust seal" is developed as another approach to trust by a type of third party certification. Examples include "WebTrust", "Thawte web server certification", "TRUSTe", et al. Currently, trust seal programs only ensure that a sealed website complies with the principles on security and privacy, but do not ensure information trustworthiness.

2.3 Provenance Based Judgment

As addressed in [13], "provenance information is extremely important for determining the value and integrity of a resource". The role of provenance in the validity of data, information / knowledge has received attention by several research projects.

Buneman et al [22][23] addressed "Data Provenance" problem: the data in the Web or a database may be extracted, copied, edited, or annotated from somewhere else in the Web or databases, and this situation leads to the question regarding whether this data

is valid. From the perspective of database technology, the authors proposed a syntactic approach to computing the provenance of a piece of data generated by a database query. The data provenance considered includes two types: why-provenance (why the data is there) and where-provenance (where the data come from). The motivation for this research is very similar to the motivation for knowledge provenance. However, data provenance only targets the data returned by queries from structured or semistructured data sets; the syntactic approach of data provenance only traces the provenance of the data but does not answer the trustworthiness of data sources.

Most recently, EU provenance project [136] [70] [135] proposed an open provenance architecture to enable documentation of the process that led to the data in grid computing and e-science. This system also provide tools for creating, recording and querying provenance information.

TRELLIS (trellis.semanticweb.org)[63] is a web-based tool that enables users to annotate their information analysis or argumentation, to justify, update and share their analysis and to add their viewpoints and evaluations to other information analysis. In TRELLIS, users are able to judge the quality of a piece of information used in argumentation, by examining other users' evaluations and the usefulness of the information in other argumentation, as well as by tracing back to the provenance of the information. One of the distinguished features of TRELLIS is that the quality of a piece of information can be assessed by not only other users' evaluations but also the usefulness and provenance of the information, that is, the use and the original context of the yield of this information. However, TRELLIS does not provide a formal model to assess the quality of a piece of dependent information by considering the quality of the information it depends on. TRELLIS can be a useful tool in practice but there is little work on formalizing the judgment of information trustworthiness.

Coming from an automated reasoning perspective, KSL at Stanford [127] developed "Inference Web (IW)". IW enables information creators to register proofs with prove-

nance information in IW, and then IW is able to explain the provenance of a piece of requested knowledge. IW only provides provenance information (registered by creators) to information users, and the users make decisions whether to trust or not trust the requested knowledge. IW mainly focuses on the explanations to reasoning by providing provenance information. In IW, there is no formal models for trust judgments. In addition, IW may be suitable for only formalized information rather than various forms of web data.

Ding et al [42] proposed provenance and trust based heuristics for homeland security information search, information integration and analysis on the Semantic Web. Very interestingly, some important information could be discovered by integrating the data spread over the Semantic Web; the trustworthiness of the discovery depends on the the provenance and the corresponding trustworthiness of each piece of information used for that derivation.

Provenance tells not only what is the information source but also how this information is derived, what is the context of this information, and how this information is used. However, processing rich provenance information needs human beings' participation, so research have to trade off the range of provenance information, what human beings do, and what machines do.

2.4 Relevant Technologies

Many AI technologies and web technologies, in particular, the Semantic Web technologies, can be used to support KP. We briefly introduce these technologies in the following subsections.

2.4.1 Digital Signature and Certification

Digital signatures are an approach to enable information recipients to verify the authenticity of the information origin and data integrity [195]. PKI (Public Key Infrastructure) is a security architecture to support digital certification. Currently, there are several PKI standards. The most well known is X.509/PKIX [26], which is a hierarchically structured PKI with a root certificate authority (RCA). In a PKI of x.509 type structure, the trust is centered at the root, and then transferred hierarchically to all the users in the network via certificate authorities (CAs)[168]. Other examples of PKI standards include PGP []and SDSI/SPKI [134]. In contrast to X.509's hierarchical structure, they have unstructured frameworks, and they are issued by individuals. In particular, we are interested in PGP's certification model " web of trust" which have been discussed earlier.

XML digital signature has been developing in W3 Consortium [178]. XML signatures are digital signatures for XML data documents. XML signatures have some new features [167], e.g. it is able to sign only specific portions of the XML tree rather than the complete document, and it can sign more than one type of resources in a single signature. The development of XML digital signature will facilitate the certification in knowledge provenance.

No doubt, digital signature and digital certification play important roles in the *Web of Trust*. However, they only provide an approach to certifying an individual's identification and information integrity, but they do not determine whether this individual can be trusted.

2.4.2 The Semantic Web

Most information on the Web is designed for human consumption, so that it is very difficult to maintain the evolution and validity of information automatically by machines. The development of the Semantic Web technologies shows a promising approach to sup-

port knowledge provenance.

The Semantic Web is an extension of the current Web in which information will be represented in a machine processable or "understandable" form, so that machines are able to share and process the data on the Web [10]. The semantic web approach is to use *metadata* to describe the web data and to use commonly shared web *ontologies* to construct new ontologies. *Metadata* is "data about data", specifically, here it is "data describing Web resources". "*Ontology* is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related."[179] More specifically, ontologies are formal and explicit definition of concepts.

As first steps towards semantic web, RDF (Resource Description Framework) [177] is recommended by W3C (World-Wide Web Consortium) for representing the metadata of Web resources. RDF is a simple and general-purpose language designed to provide interoperability between applications that exchange machine-understandable information on the Web. This feature enable automated processing of web resources. Even though RDF is originally designed for representing metadata about web resources, in fact, it can be used to describe any object on the Web, for anything on the web can be regarded as web resource. RDF with digital signatures will be the key to building the 'Web of Trust' for e-commerce, collaboration, and other applications [177].

RDF uses RDF graph as a knowledge representation model. A RDF graph can be defined as a set of triples of the form $< O, A, V >$ that represents that object O has attribute A which has value V, or a statement with subject O, predicate A, and object V. RDF uses URI (Uniform Resource Identifier) to identify one thing from the others. In this way, RDF is used to represent information on the Web and to make the information exchangeable between different applications. RDFS (RDF Schema) [181] extends RDF by introducing basic facilities to describe classes, properties, instances, and constraints on the relation between classes and properties.

RDF is defined based upon XML [182]. So, any RDF data file is a XML data file

also. XML is designed as a text format extensible markup language. It can be used to assign data with semantics; It represents information in structured data; its text format makes it be a good platform-free data exchange media. Due to these features, it widely used as data file in many heterogeneous applications.

To facilitate publishing and sharing ontologies on the Web, OWL, web ontology language, is recommended by W3C to define and instantiate classes and relations. From a set of web ontologies, facts not given but entailed by the ontologies can be derived by using OWL formal semantics [179]. "OWL is a vocabulary extension of RDF [180]." In this way, an OWL ontology is an RDF graph (a set of RDF triples). OWL has three subsets: OWL Lite, OWL DL and OWL Full. Among them, OWL DL is supported with Description Logic reasoners. OWL Lite is a light version of OWL DL. OWL Full is designed to be compatible with RDF, so that any RDF graph is a OWL Full ontology.

In addition to the infrastructure to facilitate machine understandable information / knowledge representation on the web, such as RDF, RDF schema and OWL, the semantic web community mainly focus on web data integration and knowledge sharing. Regarding KP, although " web of trust" is identified as the top layer of the semantic web, except XML digital signature as an essential step towards knowledge provenance, currently no standard for knowledge provenance level exists.

2.4.3 Knowledge Representation Technologies

Various knowledge representation technologies developed in artificial intelligence provide many alternatives for knowledge representation in KP. This section reviews several directly relevant technologies.

Knowledge and Belief in Epistemic Logic

From the literature on trust, belief is the kernel element of trust. The logic of knowledge and belief has been studied in epistemic logic. Epistemic logic [130] is a particular modal

logic. Modal logic [27] is the logic of *necessity* and *possibility*. *Necessity* refers to that propositions are necessarily true (i.e. "must be true", denoted as $\Box p$, and p denotes a proposition); and *possibility* refers to that propositions are possibly true (i.e. "may be true", denoted as $\Diamond p$).

The most commonly used axioms in modal logic include [33][187]:

K : $\Box(p \supset q) \supset (\Box p \supset \Box q)$

T : $\Box p \supset p$

D : $\Box p \supset \Diamond p$

4 : $\Box p \supset \Box\Box p$

5 : $\Diamond p \supset \Box\Diamond p$

Based on these axioms, several representative modal logic systems (refer to [129][187]) are defined. Typically, the system **T** has axioms $\boldsymbol{K}$ and $\boldsymbol{T}$; the system **S4** has axioms $\boldsymbol{K}$, $\boldsymbol{T}$ and $\boldsymbol{4}$; the system **S5** has axioms $\boldsymbol{K}$, $\boldsymbol{T}$ and $\boldsymbol{5}$; the system **K45** has axioms $\boldsymbol{K}$, $\boldsymbol{4}$, and $\boldsymbol{5}$; the system **KD45** (also called **weak-S5**) has axioms $\boldsymbol{K}$, $\boldsymbol{D}$, $\boldsymbol{4}$, and **5**. These systems are widely used to represent the logic of knowledge and belief in epistemic logic.

In epistemic logic, knowledge and belief are formalized with the possible world semantics [85][109]. By the semantics, a model, M, is a triple $< W, R, V >$, where W is the set of possible worlds, R is accessibility relation, $R \supset W \times W$, and $V : W \times P \rightarrow \{T, F\}$ is a truth assignment function for each atom proposition in each possible world. In Kripke's view, the semantics of accessibility relation is as follows. $(w, w') \supset R$ means that w' is "possible relative to" w, i.e., every proposition true in w is possible in w'. Corresponding to the axioms $\boldsymbol{T}$, $\boldsymbol{D}$, $\boldsymbol{4}$, and $\boldsymbol{5}$, the accessibility relation R should be reflexive, serial, transitive, and Euclidean respectively.

In possible world semantics, proposition p is necessarily true, if the proposition is true in all accessible possible worlds, i.e.

$$< M, w > \models \Box p, iff : \forall w', (w, w') \in R \supset < M, w' > \models p \tag{2.1}$$

Proposition p is possibly true, if the proposition is true in some of the accessible

possible worlds, i.e.

$$< M, w > \models \Diamond p, iff : \exists w', (w, w') \in R \wedge < M, w' > \models p \tag{2.2}$$

In epistemic logic, knowledge is defined as the propositions necessarily true. The systems **T**, **4**, and especially **5**, are used as the logics of knowledge. Different from knowledge, belief need not to be necessarily true, so that axiom ***T*** cannot be applied to the logic systems of belief. For this reason, the systems **K45** and **KD45** (**weak-S5**) are used as the logic of belief.

From our point of view, belief is possibly true; on the other hand, a proposition possibly true is not necessarily to be believed. In other words, possibly true is a necessary condition of belief, but it is not the sufficient condition.

Situation Calculus

The situation calculus is a logic language specifically designed for representing dynamically changing worlds [150]. It works in the following way: the changing world is represented by a set of fluents. A fluent is a property (of the world) whose value is dependent on situations. In other words, a fluent dynamically changes when the situation does. The situation, in turn, changes when an action is performed by agent(s) in the world.

The situation calculus language is a sorted second order logic language [150]. We will only use the first order logic part of the situation calculus. In a sorted logical language, the universe is participated into disjoint sub-universes, and each sub-universe corresponds to a sort. Sorts are similar to the "types" in a programming language. In the situation calculus, there are four domain independent sorts: A, S, F, and D denote actions, situations, fluents and domain objects respectively.

The representation of actions, situations and fluents are further discussed as follows.

Actions Actions are represented with terms. A term is the same as its definition in the first order logic. A term can be a variable, or a function of arity n, $f(t_1, ..., t_n)$, where

t_1, ..., t_n are terms. (Note: a constant actually is a function of arity 0).

Situations Situations are represented with terms. There is an initial situation, called S_0, representing the situation in which no actions have been done. Function "$do(a, s)$" maps to the situation after doing action a in situation s.

Fluents There are two types of "fluents": (1) "*relational fluents*", refer to relations that have true values of "*true*" or "*false*"; (2) "functional fluents", refer to the functions as defined in mathematics. Functional fluents are represented as terms. For relational fluents, there are two types of representation: reified and non-reified representation [147]. In non-reified representation (used by the classical situation calculus), a fluent is represented as a predicate in the form of $f(x, s)$, where f is the name of the predicate to represent a fluent, x denotes the objects in the domain of this fluent, and s is the situation in which the fluent holds. In reified representation, a fluent is represented as a term in the form of $f(x)$, and a fluent $f(x)$ is true in situation s is represented by predicate $holds(f(x), s)$. In this way, a fluent is a term. So that a fluent may have other fluents as parameters.

The situation calculus interests us for two reasons: first, situations in the situation calculus provide a solution to formally represent the contexts of trust; secondly, trust dynamically changes with the growth of the knowledge related to trust. These two features make the situation calculus a good tool used to formalize trust.

Truth Maintenance Systems

Information dependency is a factor considered in knowledge provenance. Truth Maintenance System (TMS) is a good tool to facilitate the representation of dependency networks for derivations.

Truth Maintenance System (TMS), which was designed to maintain beliefs for general problem solving systems, was proposed by Jon Doyle in 1979 [44]. Since then, it has

become a common and widely used piece of AI technology [35]. Truth Maintenance Systems are classified in three types: Justification-based TMS (JTMS), Logic-based TMS (LTMS)[36], and Assumption-based TMS (ATMS) [34]. JTMS uses two values *in*, *out* where *in* means "having evidence to believe a proposition to be true", and *out* means "no evidence to believe a proposition to be true". LTMS uses three-valued logic *True*, *Unknown*, *False*. The relation among these values is that *in* is corresponding to *True*, and *out* is corresponding to *False* or *Unknown* (see detail in [36]). TMS facilitates the representation of dependency networks for derivations, to diagnose inconsistencies, to conduct dependency-directed backtracking, and to support default reasoning. These features make TMSs provide us useful technical approaches to represent KP models.

On the other hand, TMSs along do not work in KP, for the following reasons: first of all, similar to any other formal systems, TMSs only check the truth of facts from which a result is derived, but TMSs cannot check whether the used rules or models in the derivation are appropriate or not. It is this higher level of validity regarding the proper uses of models an important reason for judging the validity of information by judging the trustworthiness of the information creators; secondly, KP needs to consider that the validity of information dynamically change over time; finally, people may have a degree of belief in a continuous sense from "believed" to "not believed" for the reason of uncertainty in information validation.

2.5 Summary

In this chapter, we presented a review of the research related to knowledge provenance including information characteristics based judgments, trust based judgments and provenance based judgments. In addition, we also reviewed the technologies that support KP.

Although the criteria to judge the quality of web information have been developed

in library and information science [9][3], the information evaluation methods are human-users oriented. In other words, the evaluation process needs people's involvement to make subjective judgments against those criteria. Therefore, in order to construct automatic tools in knowledge provenance, based on the study of information quality evaluation, information validity judgments (a kernel part of information quality evaluation) need to be formalized. According to Wilson [186], "we can trust a text if it is the work of an individual or group of individuals whom we can trust." In this way, essentially, the validity of a web information may be judged in the base of the trust placed in the information sources.

A number of formalized trust models have been proposed. Most models focus on how trust is build up among entities. However, a finite number of trust relationships which each individual entity has cannot meet the needs of interactions with other unfamiliar or unknown entities on the web. As a promising remedy to this problem, social networks-based trust is receiving considerable attention. A necessary condition for trust propagation in social networks is that trust needs to be transitive. Nevertheless, is trust transitive? What types of trust are transitive and why? There are neither theories nor models found so far to answer these questions in a formal manner. Most models found so far either directly assume trust transitive or do not give a formal discussion of why trust is transitive. To answer these questions, further formalization of the semantics of trust is necessary.

In the light of the semantic web and knowledge representation technologies, it is possible to construct the logical models and web ontologies of knowledge provenance and trust, and to use these web ontologies annotating web contents to create islands of certainty in a morass of uncertain and incomplete information on the web.

Chapter 3

Static Knowledge Provenance

As stated in Chapter 1, the problem to be addressed in this thesis is how to determine the validity of information/knowledge on the Web. Knowledge Provenance addresses this problem by investigating how to model and maintain the origin and validity of knowledge.

We believe that people determine the validity of information in three basic ways: (1) direct judgment by using information users own knowledge to analyze the content of the given information; (2) indirect judgment by analyzing the characteristics of the given information; (3) indirect judgment by trust and analyzing the provenance of the given information. The first and the second approaches are domain and knowledge specific and information content related, so that it is difficult to build general purpose models. This thesis focuses on the third approach.

As given in Chapter 1, we identify four modules of KP. The focus of this chapter is on Static Knowledge Provenance. Static KP is concerned with the provenance of the knowledge that is both certain and does not change over time. Static KP provides the basic concepts and the fundamental building blocks for determining validity, on which dynamic and uncertain Knowledge Provenance are constructed.

Logically, the unit of web information to be considered in KP is a "proposition". A proposition, as defined in First Order Logic, is a declarative sentence that is either true

or false. In practice, a "proposition" in KP can refer to one or more xml elements, a sentence, a phrase, or even a whole document. However, a "proposition" is the smallest piece of text that may be annotated with provenance-related attributes. In other words, once a piece of text is defined as a proposition in KP, no propositions can be further defined within this piece of text.

Basically, any proposition has a truth value of *True* or *False*. When the truth value of a proposition cannot be determined to be true or false, the truth value is set as "Unknown". The use of "Unknown" is a simple solution to handle uncertainty in static KP.

Although the unit of web information to be considered in KP is a "proposition", which is called a *KP_prop*, this type of "propositions" are **objects** to be processed by KP models, and they are not propositions in the logical model of KP; therefore, we can use *First Order Logic* as the language to represent KP models. Those *KP_props* can be regarded as "reified" propositions. We will discuss this later in this chapter.

In order to give a formal and explicit specification for Static KP and to make it available on the web, a static KP ontology is defined in this chapter. Following the ontology development methodology of Gruninger and Fox [74], we specify static knowledge provenance ontology in four steps:

(1) Provide a motivating scenario;

(2) Define informal competency questions for which the ontology must be able to derive answers;

(3) Define the terminology (i.e. predicates);

(4) Define the axioms (i.e. semantics).

3.1 Motivating Scenarios

In the following, the underlying concepts of Static Knowledge Provenance are explored in the context of two scenarios.

Case 1: Asserted Information

Consider the proposition found on a web page that "perennial sea ice in the Arctic is melting faster than previously thought at a rate of 9 percent per decade." From a provenance perspective, there are three questions that have to be answered: 1) What is the truth value of this proposition? 2) Who asserted this proposition? 3) Should we believe the person or organization that asserted it? In this example, a further examination of the text of the web page provides the answers (www.gsfc.nasa.gov/topstory/2002/1122seaice.html): It is a true proposition, asserted by NASA, who most people believe is an authority on the subject. Question is, how can this provenance information be represented directly without having to resort to Natural Language Processing of the page?

Other examples of asserted information include assertions made by persons or organizations, statistical data and observation data such as stock quotes and weather readings issued by organizations. In addition, commonly recognized knowledge, such as scientific laws, should be treated as "asserted information". This is for the following reason. Although scientific laws are usually regarded as "derived information", the derivation of scientific laws have been validated in the past, and needn't to be validated again.

Case 2: Dependent Information

Consider the following proposition found in another web page: "The accelerated rate of reduction of perennial sea ice in the Arctic will lead to the extinction of polar bears within 100 years." This is actually two propositions composed of a premise, "The accelerated rate of reduction of perennial sea ice in the Arctic" and a conclusion, "the extinction of

polar bears within 100 years. Just as in the previous case, there are three questions that need to be answered: 1) What is the truth value of these propositions? 2) Who assign these truth values? 3) Should we believe the person or organization that asserted them? What makes this case more interesting is that answering these question is dependent upon propositions found in other web pages. There are two types of dependency occurring. First the truth of the premise is dependent on the truth of the proposition found in another web page. Secondly, the truth of the conclusion depends on the truth of the premise and upon some hidden reasoning that led to the deduction. These types of propositions are called "dependent propositions" in KP.

It is common to find information in one document reproduced in another. The reproduction of a proposition in a second document leads to an equivalence relation between the two propositions, i.e., the truth value of the two propositions are equivalent. But the relationship is also asymmetric; one proposition is a copy of the other. The copy of one proposition is classified as "equivalent information."

Furthermore, a proposition can be derived using logical deduction. Hence, the truth value of the derived proposition depends on the truth values of its antecedent propositions. This type of derived proposition is classified as "derived information".

Returning to the example, determining the provenance of the premise requires that we link, in some way, the premise to the proposition in the other web page from which it is copied. That link will also require some type of certification so that we know who created it and whether it is to be trusted. The same is true of the conclusion. Minimally, we should link it to its premise, maximally we should link it to the axioms that justify its derivation. This link would also need to be certified in a similar manner.

In practice, a proposition may be derived by applying different axioms. For example, according to the demerit point system of Ontario's Ministry of Transportation, a person may get 3 points for the following reasons: Failing to yield the right-of-way; Failing to obey a stop sign, traffic light or railway crossing signal; Going the wrong way on a

one-way road. Each may be a possible reason for a loss of points.

Derived propositions may also be dependent upon disjunctions, conjunctions and/or negations of other propositions.

From these two cases, a number of concepts required for reasoning about provenance emerge:

- Text is divided into propositions.

- An asserted proposition must have a digital signature, to prove the authenticity of the information creator(s).

- If the assertion is to be believed, then the person or organization that signed the assertion must be acceptable to (i.e. trusted by) the user of the information.

- As propositions are reused across the web, a link between where it is used and where it came from must be maintained.

- Dependencies can be simple copies, or can be the result of a reasoning process. If the latter, then premises used in the reasoning should also be identified and signed by an acceptable organization.

3.2 Informal Competency Questions

Competency questions define the scope of an ontology. In other words, assuming some type of deductive reasoning system, an application built using the ontology must be able to deduce answers to the competency questions. The following questions define the competence of the Static KP ontology.

- What truth value can this proposition be believed to have?

- Who created this proposition? Is the information creator authentic?

- Can this information creator be trusted in a field the proposition belongs to?

- Does the truth of this proposition depend on any other propositions? can these propositions be believed to be true?

3.3 Terminology

In this thesis, sorted First Order Logic [150] (pp.9) is employed to represent KP. In a sorted logical language, the universe is partitioned into disjoint sub-universes, and each sub-universe corresponds to a sort. Predicates are syntactically restricted to have arguments of certain predefined sorts. Sorts are similar to the "types" in a programming language. To formalize KP, we introduce the following sorts.

$\boldsymbol{P}$: the set of KP_props (KP_prop instances);

$\boldsymbol{L}$: the set of classes for representing different KP_props;

$\boldsymbol{E}$: the set of entities such as individuals and organizations;

$\boldsymbol{F}$: the set of knowledge fields;

$\boldsymbol{D}$: the set of domain objects such as proposition contents and digital signatures.

In the following, the predicates will be defined to represent KP propositions, the properties of KP_propositions, as well as trust relationships.

3.3.1 Proposition Taxonomy

As stated in the begining of this chapter, the unit of web information to be considered in KP is a "proposition". *KP_prop* is the most general concept used to represent "propositions" in web documents. Based on the findings in our motivating scenarios, we define different types of KP_props in table 1, in which predicate $type(x, c)$ is defined as:

$type(x, c) \subseteq \boldsymbol{P} \times \boldsymbol{L}$

where, x is an instance of class c $\in \boldsymbol{L}$. $\boldsymbol{L}$ contains the names of 11 KP_prop types.

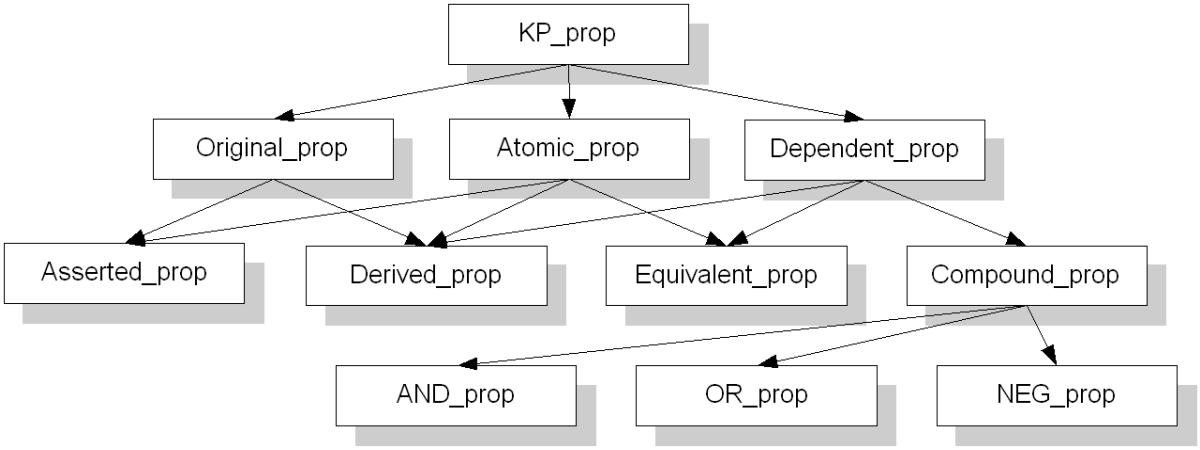

Figure 3.1: The taxonomy of KP propositions

The taxonomy of KP_props are shown in figure 3.2. In the figure, the six leaf nodes are considered as basic types.

3.3.2 Properties about Information Source and Authentication

Although the unit of web information to be considered in KP is a "proposition", which is called a *KP_prop*, this type of "propositions" are **objects** to be processed by KP models, and they are not propositions in the logical model of KP; in this way, we can use *First Order Logic* as the language to represent the properties of those "propositions" (KP_props).

Table 3.2 defines properties of KP_props related to information sources and authentication.

For any original proposition, its creator can be defined. Along with it can be defined a digital signature and the verification status of the signature. Assume that digital signature validation software provides the result of signature verification.

Table 3.1: Predicates - Proposition Types

Predicate	**Definition**
$type(x, KP\_prop)$	x is a KP_prop.
$type(x, Original\_prop)$	x is an *original proposition* created by its information creator(s). An original proposition is a piece of original work of the information creator(s), and it can be an assertion or the result of an inference process made by the information creator(s). Therefore, *Original_prop* class is further divided into two subclasses: *Asserted_prop* and *Derived_prop*.
$type(x, Dependent\_prop))$	x is a proposition whose truth value is dependent on other propositions. Dependent_prop class has 3 subclasses: Derived_prop, Equivalent_prop and Compound_prop.
$type(x, Asserted\_prop))$	x is an *asserted proposition*, which is its creators' assertion and not dependent upon any other propositions.
$type(x, Derived\_prop))$	x is a *derived proposition*, which is the result of a reasoning process, so its truth value depends on other KP_props.
$type(x, Equivalent\_prop))$	x is a *Equivalent_prop*. An Equivalent_prop is a copy of and its truth value is the same as the proposition it depends on.
$type(x, Compound\_prop))$	Compound_prop is defined to be the logical combination of its constituent propositions. A Compound-prop is divided into 3 subclasses: Neg_prop, And_prop, and Or_prop.
$type(x, Neg\_prop))$	x is the logical *negation* of the proposition it depends on.
$type(x, And\_prop))$	x is the logical *and* of the propositions it depends on.
$type(x, Or\_prop))$	x is the logical *or* of the propositions it depends on.
$type(x, Atomic\_prop)$	x is an atomic proposition (which has proposition content), that is, it is not a compound proposition. An atomic proposition is either Asserted_prop, or Derived_prop, or Equivalent_prop.

Table 3.2: Predicates - Information Sources and Authentication

Predicate	**Definition**
$has\_infoCreator(x, c)$	$\subseteq \boldsymbol{P} \times \boldsymbol{E}$ KP_prop x has information creator c. An infoCreator may be either an author or a publisher.
$has\_author(x, c)$ $has\_publisher(x, c)$	$\subseteq \boldsymbol{P} \times \boldsymbol{E}$ KP_prop x has author (or publisher) c.
$has\_signature(x, s)$	$\subseteq \boldsymbol{P} \times \boldsymbol{D}$ The proposition x has a digital signature s.
$valid\_sig(x, s, c, a)$	$\subseteq \boldsymbol{P} \times \boldsymbol{D} \times \boldsymbol{E} \times \boldsymbol{E}$ From the perspective of provenance requester a, x has valid digital signature s signed by c. In other words, the digital signature of x has been validated by a. This predicate corresponds to an external digital signature validation process. This process returns "True" if the digital signature is validated successfully; otherwise "False".
$valid\_webPub(x, p, a)$	$\subseteq \boldsymbol{P} \times \boldsymbol{E} \times \boldsymbol{E}$ From the perspective of provenance requester a, KP_prop x is a valid web publication of p, that is, x has a URL used by p for web publication. This predicate corresponds to an external process that checks the validity of web publication and returns "True" if valid otherwise "False".
$has\_authentic\_source(x, s, a)$	$\subseteq \boldsymbol{P} \times \boldsymbol{E} \times \boldsymbol{E}$ From the perspective of provenance requester a, KP_prop x has authenticated information source s. In other words, the information creator of x has been authenticated by a, which means either the digital signature of x has been validated by a or x is a valid web publication of s.

3.3.3 Truth Values and Other Properties

KP_props has two types of truth values: assigned truth value, which is the truth value claimed by information creators; believed truth value, which is the truth value inferred and believed by information users. Only Original_props need assigned truth value; but every KP_prop has believed truth value. For a KP_prop, different information users may have different believed truth values. Basically, any proposition has a truth value of *True* or *False*. When the believed truth value of a proposition cannot be determined to be true or false, the believed truth value is set as "Unknown". The use of "Unknown" is a simple solution to handle uncertainty in static KP.

Table 3.3 defines the predicates for depicting the truth values and other properties of a KP_prop.

3.3.4 Trust Relationships

From literature review, we know that the cognitive authority of a text can be determined by the cognitive authority of the information creators [186] . According to this fact, KP has a basic rule to determine the validity of a proposition as follows. If an information creator is trusted in a field, then any proposition created by the creator in the field is believed. We formally define the semantics of trust in chapter 5. Based on that formal semantics, in this chapter, several trust related predicates are directly defined and used. Trust related predicates are defined in table 3.4.

3.3.5 Other Predicates, Function and Symbols

Other predicates and function to be used in KP are defined in table 3.5. Symbols used to represent logical systems in KP are given in table 3.6.

Regarding predicate $equivalent\_to(c, k)$, in implementation, for different form of proposition content (such as xml or xhtml data), this predicate is implemented as a program in

Table 3.3: Predicates: Properties of Propositions

Predicate	**Definition**
$assigned\_truth\_value(x, v)$	$\subseteq \boldsymbol{P} \times \{True, False\}$ Proposition x has a truth value v assigned by proposition creator. v may be one of "True" or "False".
$believed\_truth\_value(a, x, v)$	$\subseteq \boldsymbol{E} \times \boldsymbol{P} \times \{True, False, Unknown\}$ Agent a (representing provenance requester) believes that proposition x has a truth value v. v may be one of "True", "False", or "Unknown".
$prop\_content(x, c)$	$\subseteq \boldsymbol{P} \times \boldsymbol{D}$ c is the content of a atomic_prop x. In html files, the content of a proposition usually is a string; in xml files, the content of a proposition can be one or more xml elements.
$in\_field(x, f)$	$\subseteq \boldsymbol{P} \times \boldsymbol{F}$ Proposition x is in knowledge field f.
$is\_dependent\_on(x, y))$	$\subseteq \boldsymbol{P} \times \boldsymbol{P}$ Proposition x is dependent on proposition y. In other words, the truth value of proposition x is dependent on the truth value of proposition y. In this thesis, x is called dependent proposition, and y is called support proposition.
$has\_ancestor(x, y))$	Proposition x has ancestor y, iff x is dependent on y, or x is dependent on z, and z has ancestor y.

Table 3.4: Predicates: Trust-related

Predicate	Definition
$trusted\_in(a, c, f)$	$\subseteq \boldsymbol{E} \times \boldsymbol{E} \times \boldsymbol{F}$ Provenance requester (or agent) a trusts information creator c on producing information in knowledge field f. This predicate corresponds to an external process to make trust judgment by giving trustor a, trustee c, and field f. If the trust relationship holds, the process return True; otherwise return False.
$believed(x, a)$	$\subseteq \boldsymbol{P} \times \boldsymbol{E}$ Proposition x is believed by agent a, that is, a believes the truth of x which is given by the proposition's creator(s).

different way. For example, if c and k are xml data and they have the same DOM tree, then they are equivalent to each other. The simplest but very limited implementation is that c is the same string as k.

A ground predicate is the predicate in which all variables are bound to individuals (constants) in the domain of discussion.

These symbols will be defined in their contexts of discussion.

3.3.6 KP_props and Their Properties

In table 3.7, the first row lists the types of KP_prop such as "Asserted" for $Asserted\_prop$, and "Derived" for $Derived\_prop$; the first column lists the properties a KP_prop may have; and what properties each type of KP_props has are marked with "$\surd$". The properties marked with "*" are primary properties, called attributes hereafter, and other properties can be derived from these attributes.

Table 3.5: Other Predicates and Function

Predicate/Function	**Definition**
$subClassOf(x, y)$	$\subseteq \boldsymbol{L} \times \boldsymbol{L}$ This is a predicate to represent that class x is a sub-class of class y.
$equivalent\_to(c, k)$	$\subseteq \boldsymbol{D} \times \boldsymbol{D}$ This is a predicate to represent that proposition content c is equivalent to proposition content k. If content c is equivalent to content k, this predicate returns true; otherwise returns false.
$neg(x)$	$\{True, False, Unknown\} \rightarrow \{True, False, Unknown\}$ This is a function to mimic logical operator $\neg$. In other words, $neg(x)$ is a term. The function is defined as follows: *neg(True) = False; neg(False) = True; and when x is anything else rather than True and False (actually, the only the possible legal value is Unknown), neg(x) = Unknown.*

Table 3.6: Symbols Representing Logical Systems

$\mathcal{T}_{KP1}$	denotes the set of axioms and theorems for static KP ontology.
$\mathcal{T}_{KP1,KB}$	$\subset \mathcal{T}_{KP1}$ denotes the set of axioms regarding the constraints on the properties of different types of KP_props.
$\mathcal{KB}_{KP1,rules}$	$\subset \mathcal{T}_{KP1}$ denotes the set of axioms and theorems as rules for provenance reasoning.
$\mathcal{KB}_{KP1,facts}$	denotes a set of ground predicates representing the properties of KP_props related to answering a provenance request.

3.4 Axioms

In this section, we use FOL (First Order Logic) to define and axiomatize static KP ontology, which defines *KP_props* and inference rules for deriving the believed truth value of a *KP_prop*.

In this thesis, we follow the convention that all unbound variables are universally quantified in the largest scope, terms starting with uppercase are constants and terms starting with lowercase are variables.

The contents of this section are organized as follows: first, the logic of static KP is described in English; then, the axioms for proposition taxonomy are defined; the axioms about information sources and how to determine the authenticity of KP_props are given; since KP determines the believed truth value of a proposition based on the trust placed in the proposition creator(s), we define axioms to specify the meaning of trust and belief; then, based on these formal semantics, we give the rules to infer the believed truth values of each type of propositions.

Table 3.7: KP_props and Their Properties

KP_prop:	Asserted	Derived	Equivalent	Compound
propositoin_content*	√	√	√	
in_field*	√	√		
is_dependent_on*		√	√	√
has_author*	√	√		
has_publisher*	√	√		
has_infoCreator	√	√		
has_sig*	√	√		
valid_sig	√	√		
valid_webPub	√	√		
has_authentic_source	√	√		
believed	√	√		
assigned_truth_value*	√	√		
believed_truth_value	√	√	√	√

3.4.1 The Object Logic in Static KP

As stated earlier, the information unit considered in KP is "proposition", called *KP_prop*; the believed truth value of a KP_prop can be true, false, or unknown; furthermore, KP needs to consider the logical relations among those KP_props. Therefore, a 3-valued propositional logic comprising of KP_props needs to be represented in static KP ontology. This object logic to be represented in KP is described as follows.

- An *Asserted_prop* is a *KP_prop*;
- a *Derived_prop* is a *KP_prop*;
- an *Equivalent_prop* is a *KP_prop*;
- if p is a *KP_prop* and q is a *KP_prop*, $p \dot{\wedge} q$ [1] is a *KP_prop*, which is expressed with a *And_prop*;
- if p is an *KP_prop* and q is a *KP_prop*, $p \dot{\vee} q$ is a *KP_prop*, which is expressed with an *Or_prop*;
- if p is a *KP_prop*, $\dot{\neg} p$ is a *KP_prop*, which is expressed with a *Neg_prop*.

The above six types of KP_props are elementary, so called "basic types". Furthermore, *Asserted_prop*, *Derived_prop*, and *Equivalent_prop* are atomic *KP_prop*, so called *atomic_prop*; *And_prop*, *Or_prop*, *Neg_prop* are compound, so called *Compound_prop*.

Each *Asserted_prop* or *Derived_prop* has an *assigned truth value*, assigned by information creator; every *KP_prop* has a *believed truth value*, believed by a provenance requester. An *assigned truth value* must be either True or False; a *believed truth value* can be *True*, *False* or *Unknown*. "Unknown" is introduced for the situations in which the provenance requester cannot determine the truth of a proposition being *True* or *False*.

[1] In the object logic, operator $\dot{\wedge}$ denotes logical conjunction; similarly, $\dot{\vee}$ denotes logical disjunction; $\dot{\neg}$ denotes logical negation.

Table 3.8: Truth Table of Logical Operators

$\boldsymbol{p}$	$\boldsymbol{q}$	$p\dot{\wedge}q$	$p\dot{\vee}q$	$\dot{\neg}p$
T	T	T	T	F
T	F	F	T	F
T	U	U	T	F
F	F	F	F	T
F	U	F	U	T
U	U	U	U	U

The *believed truth value* of an *Asserted_prop* is determined by (1) the *assigned truth value* of this proposition, and (2) whether this proposition is believed by the provenance requester. A proposition is believed if its creator is trusted by the provenance requester in a field covering that proposition.

The *believed truth value* of an *Derived_prop* is determined by (1) the *assigned truth value* of this proposition, (2) whether this proposition is believed by the provenance requester, and (3) the *believed truth value* of the support proposition which this proposition is dependent on. Relation "*is_dependent_on*" means that if *KP_prop p* is dependent on *KP_prop q*, the *believed truth value* of *q* needs to ba taken into account in determining the *believed truth value* of *p*.

The *believed truth value* of an *Equivalent_prop* is equivalent to the *believed truth value* of the support proposition which this proposition is dependent on, if the contents of these two proposition are equivalent.

The *believed truth value* of a compound proposition is determined by the truth table of Kleene's 3-valued logic [103] as shown in table 7.

In the following, we use FOL as language to elaborate this 3-valued propositional logic of *KP_props*. In our formalization, **each *KP_prop* is handled as an object**, in other words, *KP_props* are individuals in FOL rather than propositions. In this way,

we can easily use FOL to represent the *believed truth value* and *assigned truth value*, as well as other properties of *KP_props*. Those *KP_props* can be regarded as "reified" propositions [164] (pp.37). However, there is a slight difference between "object" and "reified proposition". A "reified" proposition is a term, and the only concerned property of it is truth value; an object representing a *KP_prop* has more concerned properties related to the provenance of that proposition.

3.4.2 Domain Closure

As discussed in the terminology section, the universe or domain of discussion in KP is divided into 5 sub-universes. Domain $\boldsymbol{L}$, the set of classes for representing different KP_props, comprises the following types.

$$\begin{aligned}\boldsymbol{L} = \{&KP\_prop,\ Atomic\_prop,\ Original\_prop,\\ &Asserted\_prop,\ Dependent\_prop,\ Derived\_prop,\\ &Compound\_prop,\ And\_prop,\ Or\_prop,\ Neg\_prop\}\end{aligned}$$

For a specific provenance request, KP considers only the questioned KP_prop and its ancestors (the propositions it directly or indirectly depends on). From a practical perspective, we assume the length of any dependency path is finite. Thus, only a finite number of KP_props are concerned for KP to answer a provenance requester. For this reason, assume that domain $\boldsymbol{P}$, the set of KP_props (KP_prop instances) considered, comprises a finite number of propositions, i.e.

$$\mathbf{P} = \{P_1, ..., P_n\}.$$

Corresponding to these propositions, other 3 domains are determined and each of them also has a finite number of individuals. Assume $\boldsymbol{E}$, the set of entities appearing as information creators and the provenance requester,

$$\mathbf{E} = \{E_1, ..., E_m\};$$

domain $\boldsymbol{F}$, the set of knowledge fields which those propositions belong to,

$$\mathbf{F} = \{F_1, ..., F_k\};$$

and domain $\boldsymbol{D}$, the set of proposition contents and digital signatures of those propositions,

$$\mathbf{D} = \{D_1, ..., D_l\}.$$

The following axiom schema states that the above individual s are all the constants in KP's knowledge base.

Domain Closure Axiom (DCA):

$$\begin{aligned}(\forall x)((x = P_1) \vee ... \vee (x = P_n) \vee (x = E_1) \vee ... \vee (x = E_m) \\ \vee (x = F_1) \vee ... \vee (x = F_k) \vee (x = D_1) \vee ... \vee (x = D_l) \\ \vee (x = KP\_prop) \vee (x = Atomic\_prop) \vee (x = Original\_prop) \\ \vee (x = Asserted\_prop) \vee (x = Derived\_prop) \\ \vee (x = Dependent\_prop) \vee (x = Compound\_prop) \\ \vee (x = And\_prop) \vee (x = Or\_prop) \vee (x = Neg\_prop)) \quad (3.1)\end{aligned}$$

3.4.3 Proposition Taxonomy

First, two general (not specific to KP) axioms about the relationships between instance, class and subclass are given as follows.

Axiom A-1:

$$type(x, y) \wedge subClassOf(y, z) \supset type(x, z). \quad (3.2)$$

Axiom A-2:

$$subClassOf(x, y) \wedge subClassOf(y, z) \supset subClassOf(x, z). \quad (3.3)$$

We have identified 12 types of KP propositions as shown in 3.2. Each type of propositions is handled as a class. The taxonomy of these classes are defined by the following axiom.

Axiom KP-0:

$$subClassOf(Atomic\_prop, KP\_prop). \tag{3.4}$$

$$subClassOf(Asserted\_prop, Atomic\_prop). \tag{3.5}$$

$$subClassOf(Derived\_prop, Atomic\_prop). \tag{3.6}$$

$$subClassOf(Equivalent\_prop, Atomic\_prop). \tag{3.7}$$

$$subClassOf(Original\_prop, KP\_prop). \tag{3.8}$$

$$subClassOf(Asserted\_prop, Original\_prop). \tag{3.9}$$

$$subClassOf(Derived\_prop, Original\_prop). \tag{3.10}$$

$$subClassOf(Dependant\_prop, KP\_prop). \tag{3.11}$$

$$subClassOf(Derived\_prop, Dependant\_prop). \tag{3.12}$$

$$subClassOf(Equivalent\_prop, Dependant\_prop). \tag{3.13}$$

$$subClassOf(Compound\_prop, Dependant\_prop). \tag{3.14}$$

$$subClassOf(And\_prop, Compound\_prop). \tag{3.15}$$

$$subClassOf(Or\_prop, Compound\_prop). \tag{3.16}$$

$$subClassOf(Neg\_prop, Compound\_prop). \tag{3.17}$$

3.4.4 Information Sources and Authentication

In KP, any author or publisher of an Original_prop is an information creator of the proposition.

Axiom KP-1:

$$type(x, Original\_prop) \supset$$

$$((has\_author(x, c) \vee has\_publisher(x, c)) \supset has\_infoCreator(x, c)). \tag{3.18}$$

Strictly, the proof of the authenticity of a proposition's authorship needs digital signatures and certification. Consider that many organizations publish information in their

web pages without digital signatures. This thesis relaxes the conditions of authenticity to validate either the digital signature or the validity of the url of the questioned proposition. However, in a real implementation, information users could turn this option off.

The following axiom defines the semantics of that a *KP_prop* has authentic source. For any *KP_prop*, it has an authenticated information creator, if: either (1) the signature of this proposition signed by this information creator is validated; or (2)the proposition is a valid web publication of this information creator. For example, the url of the proposition shows that it is in the official web pages of the information creator.

Axiom KP-2:

$$type(x, Original\_prop) \supset (has\_infoCreator(x, c) \wedge (has\_signature(x, s) \wedge valid\_sig(x, s, c, a) \vee valid\_webPub(x, c, a)) \supset has\_authentic\_source(x, c, a)). \quad (3.19)$$

The above axiom can be rewritten as two rules to infer whether a *KP_prop* has an authenticated information creator.

Rule KP-2a:

$$type(x, Original\_prop) \wedge has\_signature(x, s) \wedge has\_infoCreator(x, c) \wedge valid\_sig(x, s, c, a) \supset has\_authentic\_source(x, c, a). \quad (3.20)$$

Rule KP-2b:

$$type(x, Original\_prop) \wedge has\_infoCreator(x, c) \wedge valid\_webPub(x, c, a) \supset has\_authentic\_source(x, c, a). \quad (3.21)$$

3.4.5 Semantics of Trust and Belief

In knowledge provenance, *trust* means that the provenance requester (an information user, the trustor in the trust) believes the information created by the information creator (trustee) to be true in a specific knowledge field (the context of trust), which is formally described by the following axiom.

Axiom KP-3 (Formal semantics of trust in KP):

$$trusted\_in(a, c, f) \supset (type(x, Original\_prop) \wedge has\_authentic\_source(x, c, a) \wedge in\_field(x, f) \supset believed(x, a)). \quad (3.22)$$

Directly from the semantics of *trust* (axiom KP-3), we have the following theorem. In KP, an original proposition is *believed* by an agent, if: the proposition has an authentic information source which is trusted by this agent in a field which the proposition belongs to.

Theorem KP-1:

$$type(x, Original\_prop) \wedge has\_authentic\_source(x, c, a) \wedge in\_field(x, f) \wedge trusted\_in(a, c, f) \supset believed(x, a). \quad (3.23)$$

The proof of this theorem is given in Appendix C.

This theorem can be used to infer that a KP_prop is believed by an agent when the agent trusts one of the information sources of the proposition.

Different from epistemic logic, in which belief is represented as an operator applied to a proposition and the the logics of belief are the theme of study, in this thesis, we represent a KP_prop as an object rather then a proposition, and then we represent *belief*

with predicate $believed(x, a)$, which means proposition x is believed by agent a. Our purpose to introduce *belief* is to determine the believed truth value of an KP_prop. For this purpose, the meanings (or effects) of *belief* in KP is discussed in the following two axioms. The general meaning of *belief* will be discussed in Chapter 5.

"*Belief*" in an asserted proposition means that the provenance requester believes the truth of the proposition in question as its creator assigned. This semantics is formally expressed as the following axiom:

Axiom KP-4 (Formal semantics of *belief* in an asserted proposition):

$$type(x, Asserted\_prop) \wedge believed(x, a) \supset (assigned\_truth\_value(x, v) \supset believed\_truth\_value(a, x, v)). \tag{3.24}$$

Regarding the semantics of *belief* in a derived proposition, since a derived proposition is the result of an inference process, similar to asserted proposition, what is believed in a derived proposition is only the work of information creator(s), i.e. the inference and the result; but the premise(s) used to infer this proposition is unnecessary to be covered by this belief. In other words, *belief* in a derived proposition is conditional, or a derived proposition is conditionally believed. The condition for this belief is that the proposition(s) which this derived proposition depends on is true.

In summary, the semantics of *belief* in an derived proposition is that the provenance requester believes the truth of the proposition in question as its creator assigned in the condition of that the support proposition of this derived proposition is true. This semantics is formally expressed as axiom KP-5.

Axiom KP-5 (Formal semantics of *belief* in a derived proposition):

$$type(x, Derived\_prop) \wedge is\_dependent\_on(x, y) \wedge believed(x, a) \supset$$
$$((believed\_truth\_value(a, y, True) \supset$$
$$(assigned\_truth\_value(x, v) \supset believed\_truth\_value(a, x, v))). \quad (3.25)$$

3.4.6 Believed Truth Values

For a different agent a, the believed truth value of a KP_prop may be different, because different agent usually has different trust relationships.

Different types of *KP_props* have different rules to infer the believed truth value. In the following, we discuss how to infer the believed truth value of each type of KP_props.

Asserted Propositions

Directly from the semantics of *belief* (axiom KP-4), we have the following theorem. An asserted-proposition has its truth value as assigned, if the asserted_prop is believed by the agent making the provenance request.

Theorem KP-2:

$$type(x, Asserted\_prop)$$
$$\wedge believed(x, a) \wedge assigned\_truth\_value(x, v)$$
$$\supset believed\_truth\_value(a, x, v). \quad (3.26)$$

The proof of this theorem is given in Appendix C.

This theorem provides the rule for inferring the believed truth value of Asserted_props.

Derived Propositions

A *Derived_prop* is the result of an inference process, so its truth value depends on the support propositions, and the inference rules or the theories used in the reasoning, from

which this result is derived.

In this KP ontology, we treat all the propositions which a *Derived_prop* depends on as one compound proposition (And_prop). In this way, a *Derived_prop* has one and only one dependency-link pointing to that compound proposition, called this *Derived_prop*'s support proposition.

Theoretically, a *Derived_prop* should have all the propositions from which it is derived included in its support proposition, and the logical relation between them is "entail". For example, *Derived_prop B* has dependency-link pointing to *KP_prop A*, which means that A entails B, i.e. $A \supset B$.

However, in real applications, it might be impractical to require to link to all knowledge used to derive a proposition. For this reason, we suggest that only those critical propositions which a *Derived_prop* depends on are included in this *Derived_prop*'s support proposition. By this way, the problem coming up is that the support proposition of a *Derived_prop* is no longer sufficient for this *Derived_prop*. This problem is one of the reasons for using trust in information creator(s) in KP.

Another reason for using trust is as follows. a *Derived_prop* is the result of an inference process which could be a very complex process, even may be not formally provable, for example, in humanity or art fields. Therefore, the judgment about the validity of such inference needs to use trust placed in the information creator(s).

From axiom KP-5 regarding the semantics of the belief in a derived proposition, we have the following theorem: the believed truth value of a derived proposition is "True" or "False" as assigned by its author, if it is believed and its support KP_prop is "True".

Theorem KP-3:

$$type(x, Derived\_prop) \wedge assigned\_truth\_value(x, v) \wedge believed(x, a) \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, True) \supset believed\_truth\_value(a, x, v). \quad (3.27)$$

The proof of this theorem is given in Appendix C.

This theorem provides the rule for inferring the believed truth value of Derived_props.

Equivalent Propositions

The believed truth value of an equivalent-proposition is the same as the believed truth value of the proposition it depends on. A valid equivalent proposition should have the content that is equivalent to the content of the support proposition.

As discussed in terminology, for different forms of proposition content (such as xml or xhtml data), the meaning of "equivalent" may be different. For example, two xml elements are equivalent if they have the same DOM tree.

Axiom KP-6:

$$type(x, Equivalent\_prop) \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, v) \wedge prop\_content(x, c_1) \wedge prop\_content(y, c_2) \wedge equivalent\_to(c_1, c_2) \supset believed\_truth\_value(a, x, v). \quad (3.28)$$

Compound Propositions

The believed truth value of a negative-proposition is the negation of the believed truth value of the KP_prop it is dependent on.

Axiom KP-7:

$$type(x, Neg\_prop) \\ \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, v) \\ \supset believed\_truth\_value(a, x, neg(v)). \quad (3.29)$$

Here, neg(v) is a function as defined in terminology, which is a function to mimic logical operator $\neg$.

The truth value of an And_prop is "True", if and only if: all its support KP_props are "True"; The truth value of an And_prop is "False", if and only if: at least one of its support KP_props is "False".

Axiom KP-8a:

$$type(x, And\_prop) \\ \wedge is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge y_1 \neq y_2 \\ \wedge believed\_truth\_value(a, y_1, True)) \wedge believed\_truth\_value(a, y_2, True)) \\ \supset believed\_truth\_value(a, x, True). \quad (3.30)$$

Axiom KP-8b:

$$type(x, And\_prop) \\ \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, False) \\ \supset believed\_truth\_value(a, x, False). \quad (3.31)$$

The believed truth value of an or_prop is "True", if and only if: at least one of its dependency KP_props is "True"; the believed truth value of an or_proposition is "False", if and only if: all its support KP_props are "False".

Axiom KP-9a:

$$type(x, Or\_prop) \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y_1, True) \supset believed\_truth\_value(a, x, True). \quad (3.32)$$

Axiom KP-9b:

$$type(x, Or\_prop) \wedge is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge y_1 \neq y_2 \wedge believed\_truth\_value(a, y_1, False)) \wedge believed\_truth\_value(a, y_2, False)) \supset believed\_truth\_value(a, x, False). \quad (3.33)$$

Default Believed Truth Values

The believed truth value of a KP_prop is "Unknown", if either "True" or "False" cannot be derived.

Axiom KP-10:

$$type(x, KP\_prop) \wedge \neg believed\_truth\_value(a, x, True) \wedge \neg believed\_truth\_value(a, x, False) \supset believed\_truth\_value(a, x, Unknown). \quad (3.34)$$

For this axiom, any web proposition that does not contain enough provenance information for KP reasoning or even does not exist at all will have believed truth value of "Unknown".

We will discuss how to derive negative formulas later in section 4.5.

Symbol $\mathcal{KB}_{KP1,rules}$ is introduced to denote all rules to be used for provenance reasoning in static KP.

$\mathcal{KB}_{KP1,rules}$ = {DCA, axioms A_1, 2, axioms KP_0, 1, 2, theorems KP_1, 2, 3, axioms KP_6, 7, 8, 9, 10}.

3.4.7 Property Constraints

Each KP_prop has certain properties; some properties need to be defined by information creator; some others can be derived with KP axioms. In the earlier part of this section, we have discussed the axioms and theorems used to derive the provenance related properties. In this subsection, we discuss how to define the primary properties of KP_props and how to represent KP_prop instances.

As discussed in subsection 3.3.6, a different type (class) of KP_props has a different set of properties. This type of constraints can be described by cardinality, for example, each Original_prop has one and only has one assigned truth value. The following axioms describe cardinality constraints.

Axiom KP-11 states that any instance of KP_prop must be the instance of one and exactly one of the six basic types of KP_props.

Axiom KP-11:

$$
\begin{aligned}
&(type(x, Asserted\_prop) \vee type(x, Derived\_prop) \vee type(x, equivalent\_prop) \\
&\qquad \vee type(x, Neg\_prop) \vee type(x, And\_prop) \vee type(x, Or\_prop)) \\
&\wedge (type(x, Asserted\_prop) \supset \neg type(x, Derived\_prop) \wedge \neg type(x, Equivalent\_prop) \\
&\qquad \wedge \neg type(x, Neg\_prop) \wedge \neg type(x, And\_prop) \wedge \neg type(x, Or\_prop)) \\
&\wedge (type(x, Derived\_prop) \supset \neg type(x, Asserted\_prop) \wedge \neg type(x, Equivalent\_prop) \\
&\qquad \wedge \neg type(x, Neg\_prop) \wedge \neg type(x, And\_prop) \wedge \neg type(x, Or\_prop)) \\
&\wedge (type(x, Equivalent\_prop) \supset \neg type(x, Asserted\_prop) \wedge \neg type(x, Derived\_prop) \\
&\qquad \wedge \neg type(x, Neg\_prop) \wedge \neg type(x, And\_prop) \wedge \neg type(x, Or\_prop))
\end{aligned}
$$

$$\wedge (type(x, Neg\_prop) \supset \neg type(x, Asserted\_prop) \wedge \neg type(x, Derived\_prop)$$
$$\wedge \neg type(x, Equivalent\_prop) \wedge \neg type(x, And\_prop) \wedge \neg type(x, Or\_prop))$$
$$\wedge (type(x, And\_prop) \supset \neg type(x, Asserted\_prop) \wedge \neg type(x, Derived\_prop)$$
$$\wedge \neg type(x, Equivalent\_prop) \wedge \neg type(x, Neg\_prop) \wedge \neg type(x, Or\_prop))$$
$$\wedge (type(x, Or\_prop) \supset \neg type(x, Asserted\_prop) \wedge \neg type(x, Derived\_prop)$$
$$\wedge \neg type(x, Equivalent\_prop) \wedge \neg type(x, And\_prop) \wedge \neg type(x, Neg\_prop)) \quad (3.35)$$

Axiom KP-12 defines property constraints on an Original_prop. An Original_prop's assigned truth value is either "True" or "False"; has at least one information creator; belongs to at least one knowledge field.

Axiom KP-12:

$$type(x, Original\_prop) \supset$$
$$((assigned\_truth\_value(x, True) \vee assigned\_truth\_value(x, False))$$
$$\wedge \neg(assigned\_truth\_value(x, True) \wedge assigned\_truth\_value(x, False))). \quad (3.36)$$

$$type(x, Original\_prop) \supset (\exists c, has\_infoCreator(x, c)) \quad (3.37)$$

$$type(x, Original\_prop) \supset (\exists f, in\_field(x, f)) \quad (3.38)$$

Note that in an implementation, the default assigned truth value is "True". In other words, if there is no assigned truth value is specified, then the assigned truth value is "True".

Axiom KP-13 states that each *Atomic_prop* has and only has one *prop_content*.

Axiom KP-13:

$$type(x, Atomic\_prop) \supset$$
$$((\exists c, prop\_content(x, c))$$
$$\wedge (prop\_content(x, c_1) \wedge prop\_content(x, c_2) \supset c_1 = c_2)). \quad (3.39)$$

Axiom KP-14 defines property constraints on *Dependent_prop* (and its sub-classes). Each *Dependent_prop* is dependent on at least one proposition that does not has this *Dependent_prop* as an ancestor (Note that predicate has_ancestor is defined in axiom KP-15); a *Derived_prop*, *Equivalent_prop* or *Neg_prop* is dependent on one and only one *KP_prop*; an And_prop or Or_prop must be dependent on two support propositions.

Axiom KP-14:

$$type(x, Dependent\_prop) \supset$$
$$(\exists y, is\_dependent\_on(x, y) \wedge \neg has\_ancestor(y, x)) \quad (3.40)$$

$$type(x, Derived\_prop) \supset$$
$$(is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \supset y_1 = y_2). \quad (3.41)$$

$$type(x, Equivalent\_prop) \supset$$
$$(is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \supset y_1 = y_2). \quad (3.42)$$

$$type(x, Neg\_prop) \supset$$
$$(is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \supset y_1 = y_2). \quad (3.43)$$

$$type(x, And\_prop) \supset$$
$$(\exists y_1, y_2)\ (is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge y_1 \neq y_2$$
$$\wedge (is\_dependent\_on(x, y_3) \supset (y_3 = y_1 \vee y_3 = y_2)). \quad (3.44)$$

$$type(x, Or\_prop) \supset$$
$$(\exists y_1, y_2)\ (is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge y_1 \neq y_2$$
$$\wedge (is\_dependent\_on(x, y_3) \supset (y_3 = y_1 \vee y_3 = y_2)). \quad (3.45)$$

A *Derived_prop* may be dependent on multiple support propositions. In such case, logically equivalently, this *Derived_prop* is regarded as dependent on the conjunction of all those multiple support propositions.

An And_prop or Or_prop must be dependent on two support propositions. The order of those support proposition does not have any effect on the result of computing, because operator $\dot{\wedge}$ and $\dot{\vee}$ are associative.

The *is_dependent_on* relation represents the immediate dependency relation. The following axiom defines predicate *has_ancestor* to include indirect dependency relations.

Axiom KP-15:

$$has\_ancestor(x, z) \equiv$$
$$(is\_dependent\_on(x, z) \vee (\exists y)(is\_dependent\_on(x, y) \wedge has\_ancestor(y, z))). \quad (3.46)$$

If a proposition has an ancestor, this proposition either directly or indirectly depends on its ancestor.

Let $\mathcal{T}_{KP1,KB}$ denote the set of axioms regarding property constraints or cardinality, that is,

$\mathcal{T}_{KP1,KB}$ = {Axioms KP_11,12,13, 14 & 15}.

Now, we discuss the representation of the KP_prop instances. A symbol, $\mathcal{KB}_{KP1,facts}$, is introduced to represent a finite set of grounded predicates that define all KP_prop instances and their primary properties (defined properties).

For different provenance questions, the individuals in domains $\boldsymbol{P}$, $\boldsymbol{E}$, $\boldsymbol{F}$, and $\boldsymbol{D}$ may be different, and the given facts about related KP_prop instances, i.e. $\mathcal{KB}_{KP1,facts}$, are also different. In other words, $\mathcal{KB}_{KP1,facts}$ is problem specific. However, $\mathcal{KB}_{KP1,facts}$ must satisfy $\mathcal{T}_{KP1,KB}$. This constraint leads to the following axiom.

Axiom KP-16:

$$\mathcal{KB}_{KP1,facts} \supset \mathcal{T}_{KP1,KB} \quad (3.47)$$

Let

$$\mathcal{T}_{KP1,KB}{}^{+} = \mathcal{T}_{KP1,KB} \cup \{AxiomKP\_16\}.$$

By this axiom, the axioms in $\mathcal{T}_{KP1,KB}$ regarding cardinality define the structure of the instance definition of each type of KP_props.

Since Asserted_prop is a subclass of Original_prop and a subclass of Atomic_prop separately, by axioms KP-11 & 12, each Asserted_prop instance has the following form of atomic formulas in $\mathcal{KB}_{KP1,facts}$.

$$type(P, Asserted\_prop) \tag{3.48}$$

$$assigned\_truth\_value(P, True) \tag{3.49}$$

$$has\_author(P, A_1) \tag{3.50}$$

$$... \tag{3.51}$$

$$has\_author(P, A_{m_P}) \tag{3.52}$$

$$has\_publisher(P, B_1) \tag{3.53}$$

$$... \tag{3.54}$$

$$has\_publisher(P, B_{n_P}) \tag{3.55}$$

$$has\_signature(P, S) \tag{3.56}$$

$$prop\_content(P, C) \tag{3.57}$$

$$in\_field(P, F_1) \tag{3.58}$$

$$... \tag{3.59}$$

$$in\_field(P, F_k) \tag{3.60}$$

Note, if P has assigned truth value of False, the formula

$$assigned\_truth\_value(P, True)$$

should be replaced with

$$assigned\_truth\_value(P, False)$$

Similarly, each Derived_prop instance, which is also an instance of Original_prop, Atomic_prop and Dependent_prop, has the following form of atomic formulas in $\mathcal{KB}_{KP1,facts}$.

$$type(P, Derived\_prop) \tag{3.61}$$

$$assigned\_truth\_value(P, True) \tag{3.62}$$

$$has\_author(P, A_1) \tag{3.63}$$

$$... \tag{3.64}$$

$$has\_author(P, A_{m_P}) \tag{3.65}$$

$$has\_publisher(P, B_1) \tag{3.66}$$

$$... \tag{3.67}$$

$$has\_publisher(P, B_{n_P}) \tag{3.68}$$

$$has\_signature(P, S) \tag{3.69}$$

$$prop\_content(P, C) \tag{3.70}$$

$$in\_field(P, F_1) \tag{3.71}$$

$$... \tag{3.72}$$

$$in\_field(P, F_k) \tag{3.73}$$

$$is\_dependent\_on(P, Q) \tag{3.74}$$

Each Equivalent_prop, which is also an instance of Atomic_prop, has the following form of atomic formulas.

$$type(P, Equivalent\_prop) \tag{3.75}$$

$$prop\_content(P, C) \tag{3.76}$$

Each Neg_prop instance, which is also an instance of Dependent_prop, has the following form of atomic formulas.

$$type(P, Neg\_prop) \tag{3.77}$$

$$is\_dependent\_on(P, Q) \tag{3.78}$$

Each And_prop (or Or_prop) instance, which is also an instance of Dependent_prop, has the following form of atomic formulas.

$$type(P, And\_prop) \tag{3.79}$$

$$is\_dependent\_on(P, Q_1) is\_dependent\_on(P, Q_2) \tag{3.80}$$

3.5 Reasoning with Negation

$\mathcal{KB}_{KP1,facts}$, the facts of KP_props related to answer a specific provenance question, and $\mathcal{KB}_{KP1,rules}$, the rules used for provenance reasoning, form a knowledge base, which is represented as symbol $\mathcal{KB}_{KP1}$.

$$\mathcal{KB}_{KP1} = \mathcal{KB}_{KP1,rules} \cup \mathcal{KB}_{KP1,facts}.$$

Knowledge base $\mathcal{KB}_{KP1}$ can infer the believed truth value and other properties of a questioned KP_prop. However, $\mathcal{KB}_{KP1}$ is only able to infer positive formulas. In other words, the knowledge base only be able to answer whether a property is true but cannot answer whether a property is false. Moreover, the use of axiom KP-10 requires the system be able to infer a predicate to be false; more importantly, the static KP ontology needs be able to answer each competence question either true or false. For this reason, we need a way to derive negative predicates.

A direct solution is to use the "Closed World Assumption" (CWA) [20]. By CWA, if an atom p is not a logical consequence of a KB, then infer $\neg p$. However, due to the undecidability of FOL, no algorithm exists to determine whether an arbitrary atom is not a logical consequence of a KB in a finite number of steps.

For this problem, a weaker form of CWA – "Negation as Failure" (NF) rule[115], can be applied to a KB containing a finite number of Horn clauses and "general program clauses", which is an extension of the form of Horn clauses to allow negative predicates appear in the body part. By NF rule, if an atom p is failed to be proved to be true by

using every clause which can be unified to p in a KB in a finite number of steps[2], then infer $\neg p$.

In KP, the knowledge base, $\mathcal{KB}_{KP1}$, consists of only Horn clauses and one "general program clause" (axiom KP-10). Note that axioms KP-11,12,13,14 & 15 in $\mathcal{T}_{KP1,KB}$ are not Horn clauses, but they are not directly used for provenance reasoning so that they are not in the $\mathcal{KB}_{KP1}$, instead they are only applied for cardinality constraints in $\mathcal{KB}_{KP1,facts}$. So, NF-rule can be applied in KP.

This solution is based on the following knowledge completeness assumption.

Knowledge Completeness Assumption (KCA): $\mathcal{KB}_{KP1,facts}$ and $\mathcal{KB}_{KP1,rules}$ are all the knowledge used to infer the provenance properties of a *KP_prop* defined in $\mathcal{KB}_{KP1,facts}$.

In KP, NF-rule is given in formal as follows.

Negation as Failure Rule (NF-rule): for any $\mathcal{KB}$, a given knowledge base in the form of general program clauses, for any atom p,

$$\mathcal{KB} \models \neg p,\ iff:\ \mathcal{KB} \nvdash_f p$$

where $\nvdash_f$ denotes a finite failure of proof by using $\mathcal{KB}$.

The set of axioms for static KP ontology using Negation as Failure can be represented as

$$\mathcal{T}_{KP1} = \mathcal{KB}_{KP1,rules} \cup {\mathcal{T}_{KP1,KB}}^{+} \cup \{KCA, NF\_rule\}.$$

In using this ontology, first $\mathcal{KB}_{KP1,facts}$, the given facts about KP_prop instances must be legal, that is, it must satisfy the axioms in $\mathcal{T}_{KP1,KB}$ regarding cardinality; secondly, in provenance reasoning, Negation as Failure is employed to infer negative literal.

Note that $\mathcal{KB}_{KP1}$ actually is stratified [116]. In particular, in axiom KP-10, we restrict that the truth of $believed\_truth\_value(a, x, Unknown)$ must be derived from the truth of $believed\_truth\_value(a, x, True)$ and $believed\_truth\_value(a, x, False)$, but axiom KP-10

[2]More accurately, p is in the finite failure set of KB. The complete account refers to [115].

cannot be used to infer $believed\_truth\_value(a, x, True)$ or $believed\_truth\_value(a, x, False)$. This stratification and Negation as Failure rule are out of FOL. Fortunately, there is an equivalent pure FOL treatment by giving the completed definition for every predicate in a KB [115](section 14). Actually, we have developed the equivalent pure FOL model in [91]. In this thesis, in order to focus our theme of KP and to make the KP model easier be able to be understood by a wider range of audience, we give our KP ontology in a simpler form of $\mathcal{T}_{KP1}$ rather than complete definitions.

3.6 Consistency and Completeness

In order to justify the proposed static KP ontology, we discuss the consistency and completeness of KP ontology.

3.6.1 Consistency

We briefly discuss the logical consistency of the static KP ontology and then mainly focus on semantic consistency.

The logical consistency (satisfiability) of the static KP ontology $\mathcal{T}_{KP1}$ (or $\mathcal{T}_{KP1}{}^*$) can be shown as follows.

First, any examples of $\mathcal{KB}_{KP1,facts}$ in the form given in subsection 3.4.7 make $\mathcal{T}_{KP1,KB}$ satisfiable;

Secondly,$\mathcal{KB}_{KP1,facts}$ and $\mathcal{KB}_{KP1,rules} - \{axiomKP-10\}$ contain only pure Horn clauses, so they are satisfiable;

Finally, axiom KP-10 is the only rule to define the truth of predicate

$$believed\_truth\_value(a, x, Unknown),$$

so if the condition is true in the model determined by the above second step, assign "True" to this predicate; otherwise assign "False". $\mathcal{T}_{KP1,KB} \cup \mathcal{KB}_{KP1,rules}$, i.e. $\mathcal{T}_{KP1}$ is satisfiable.

In the following, we discuss consistency from the perspective of the semantics of the proposed ontology. As stated earlier, static KP elaborates a 3-valued propositional logic. If a KP_prop has two (or more) different truth values, the semantics becomes inconsistent; on the other hand, if a KP_prop has one and only one truth value (either assigned or derived), then the semantics of the ontology is consistent.

Axiom KP-12 (formula (3.36)) guarantees that each Original_prop has one and only one assigned truth value of either "True" or "False".

The following theorem guarantees the semantic consistency of believed truth value.

Theorem KP-4: *By system $\mathcal{T}_{KP1}$, given $\mathcal{KB}_{KP1,facts}$ that satisfies $\mathcal{T}_{KP1,KB}$, for any provenance requester a, any KP_prop x has one and only one believed truth value of either "True", "False" or "Unknown".*

$$\begin{aligned}
&\mathcal{KB}_{KP1,rules} \models \\
&\qquad \mathcal{KB}_{KP1,facts} \supset \\
&\qquad (type(x, KP_prop) \supset \\
&\qquad ((believed_truth_value(a, x, True) \vee believed_truth_value(a, x, False) \\
&\qquad \vee believed_truth_value(a, x, Unknown)) \\
&\qquad \wedge \neg(believed_truth_value(a, x, True) \wedge believed_truth_value(a, x, False)) \\
&\qquad \wedge \neg(believed_truth_value(a, x, True) \wedge believed_truth_value(a, x, Unknown)) \\
&\qquad \wedge \neg(believed_truth_value(a, x, False) \wedge believed_truth_value(a, x, Unknown))))
\end{aligned} \tag{3.81}$$

The proof of this theorem is given in Appendix C.

3.6.2 Completeness

We discuss the completeness of the ontology in the sense of that a knowledge base constructed with the ontology can answer every competency question.

The competency questions presented earlier are discussed one by one as follows.

Q1. What truth value can this proposition be believed to have?

Given provenance requester R and questioned KP_prop X, this question can be formally represented as follows.

$$type(X, KP\_prop) \supset \exists v, believed\_truth\_value(R, X, v). \tag{3.82}$$

By theorem KP-4, we have

$$\mathcal{KB}_{KP1,rules} \models$$
$$\mathcal{KB}_{KP1,facts} \supset$$
$$(type(X, KP\_prop) \supset (\exists v, believed\_truth\_value(R, X, v))) \tag{3.83}$$

where variable v is bound to one and only one of "True", "False" or "Unknown".

$comp(\mathcal{KB}_{KP1,facts})$ contains all KP_prop instances and their primary properties related to answer this question.

Therefore, $\mathcal{T}_{KP1}$ answers this question.

Q2. Who created this proposition? Is the information creator authentic?

Given provenance requester R and questioned proposition X, this question can be formally represented as:

$$type(X, Original\_prop) \supset$$
$$(\exists c, has\_infoCreator(X, c) \wedge has\_authentic\_source(X, c, R)) \tag{3.84}$$

Given

$$type(X, Original\_prop),$$

by axiom KP-12, we have

$$\exists c, has\_infoCreator(X, c);$$

furthermore, by axiom KP-2, if there is an information creator c and a signature s, such that

$$has\_signature(X, s) \wedge valid\_sig(X, s, c, R)$$

or

$$valid\_webPub(X, c, R)$$

is true (authentication succeeds), then

$$has\_authentic\_source(X, c, R)$$

is true; otherwise, there is no c to make

$$has\_authentic\_source(X, c, R)$$

true, so the answer to this question is false.

Thus, if authentication succeeds, we have

$$\mathcal{KB}_{KP1,rules} \models$$
$$\mathcal{KB}_{KP1,facts} \supset$$
$$(type(X, Original\_prop) \supset$$
$$(\exists c, has\_infoCreator(X, c) \wedge has\_authentic\_source(X, c, R))); \quad (3.85)$$

otherwise,

$$\mathcal{KB}_{KP1,rules} \models$$
$$\mathcal{KB}_{KP1,facts} \supset$$
$$\neg(type(X, Original\_prop) \supset$$
$$(\exists c, has\_infoCreator(X, c) \wedge has\_authentic\_source(X, c, R))). \quad (3.86)$$

Therefore, $\mathcal{T}_{KP1}$ answers this question.

Q3. Can an information creator of an Original_prop be trusted in a field the proposition belongs to?

Given provenance requester R and questioned proposition X, this question can be formally represented as:

$$type(X, Original\_prop) \wedge$$
$$(\exists c, f, has\_infoCreator(X, c) \wedge in\_field(X, f) \wedge trusted\_in(R, c, f)). \quad (3.87)$$

If X is an Original_prop

$$type(X, Original\_prop),$$

by axiom KP-12, we have

$$\exists c, has\_infoCreator(X, c)$$

and

$$\exists f, in\_field(X, f).$$

If there is a c and f, such that external trust judgment process

$$trusted\_in(R, c, f)$$

returns true, then the answer to this question is true; otherwise, false.

Thus, if X is an Original_prop and trust judgment process returns true,

$$\mathcal{KB}_{KP1,rules} \models$$
$$\mathcal{KB}_{KP1,facts} \supset$$
$$(type(X, Original\_prop) \wedge$$
$$(\exists c, f, has\_infoCreator(X, c) \wedge in\_field(X, f) \wedge trusted\_in(R, c, f))); \quad (3.88)$$

otherwise,

$$\mathcal{KB}_{KP1,rules} \models$$
$$\mathcal{KB}_{KP1,facts} \supset$$
$$\neg(type(X, Original\_prop) \wedge$$
$$(\exists c, f, has\_infoCreator(X, c) \wedge in\_field(X, f) \wedge trusted\_in(R, c, f))). \quad (3.89)$$

Therefore, $\mathcal{T}_{KP1}$ answers this question.

Q4. Does the truth of this proposition depend on any other propositions? can these propositions be believed to be true?

Given provenance requester R and questioned proposition X, this question can be formally represented as:

$$type(X, Dependent\_prop)$$
$$\wedge (\forall y)(is\_dependent\_on(X, y) \supset believed\_truth\_value(R, y, True)) \quad (3.90)$$

If X is a Dependent_prop

$$type(X, Dependent\_prop),$$

by axiom KP-12, we have

$$\exists y, is\_dependent\_on(X, y).$$

By theorem KP-4, for all such y, y has one and only one believed truth value of "True", "False" or "Unknown", so

$$believed\_truth\_value(R, y, True)$$

is either true or false.

If all support propositions of X have believed truth value of "True",

$$\mathcal{KB}_{KP1,rules} \models$$
$$\mathcal{KB}_{KP1,facts} \supset$$
$$(type(X, Dependent\_prop) \wedge$$
$$(\forall y)(is\_dependent\_on(X, y) \supset believed\_truth\_value(R, y, True))); \quad (3.91)$$

otherwise,

$$\mathcal{KB}_{KP1,rules} \models$$
$$\mathcal{KB}_{KP1,facts} \supset$$
$$\neg(type(X, Dependent\_prop) \wedge$$
$$(\forall y)(is\_dependent\_on(X, y) \supset believed\_truth\_value(R, y, True))). \quad (3.92)$$

Therefore, $\mathcal{T}_{KP1}$ answers this question.

In summary, this static KP ontology can answer all predefined competency questions.

3.7 Web Implementation

To use KP, information creators need to annotate web documents with KP metadata to describe the provenance-related attributes, such as who is proposition creator and what is the premise proposition which this proposition depends on. A web browser "plugin" is expected to assist information creators to annotate their web documents; information users (provenance requesters) need to define their personalized trust relationships to tell whom they trust in what fields; a knowledge provenance reasoner will trace KP tags (KP metadata) in web documents across web pages, combining information sources and dependencies, as well as trust relationships, to deduce the origin and validity of the questioned proposition and the propositions it depends on.

3.7.1 Web Ontology of KP in OWL

From the formal KP ontology presented earlier in this chapter, we have defined Web ontology of KP by Web Ontology Language OWL[180]. The Web ontology of KP is given in Appendix A, which is also available online at *www.eil.utoronto.ca/kp*.

The classes and their associated properties are summarized as follows. In each box, the first line is the defined class, and the other lines are the properties the class has. A sub-class also has the properties of its super-class. Each property is designated as *primitive* (i.e. given by information creator) or *derived* (i.e. derived by using KP axioms). The cardinality of each property is also given.

kp:KPProp
kp:believedTruthValue (derived) cardinality = 1

kp:AtomicProp
rdfs:subClassOf kp:KPProp
kp:propContent (primitive) cardinality = 1

kp:OriginalProp
rdfs:subClassOf kp:KPProp
kp:assignedTruthValue (primitive) cardinality = 1
kp:hasAuthor (primitive) cardinality $\geq$ *0*
kp:hasPublisher (primitive) cardinality $\geq$ *0*
kp:inField (primitive) cardinality $\geq$ *1*
kp:hasInfoCreator (derived) cardinality $\geq$ *1*
kp:hasAuthenticSource (derived) cardinality $\geq$ *0*
kp:believed (derived) cardinality = 1

kp:DependentProp

rdfs:subClassOf kp:KPProp

kp:isDependentOn (primitive) cardinality ≥ 1

kp:AssertedProp

rdfs:subClassOf kp:AtomicProp

rdfs:subClassOf kp:OriginalProp

kp:DerivedProp

rdfs:subClassOf kp:AtomicProp

rdfs:subClassOf kp:OriginalProp

rdfs:subClassOf kp:DependentProp

Note: By KP ontology in FOL, the cardinality of Derived_prop should be 1; in Web ontology, we relax this constraint to allow multiple dependency. When cardinality is greater than 1, KP reasoner assumes an implicit And_prop being the conjunction of all support propositions.

kp:EquivalentProp

rdfs:subClassOf kp:AtomicProp

rdfs:subClassOf kp:DependentProp

(kp:isDependentOn (primitive) cardinality = 1)

Note: property *isDependentOn* has been defined in superclass *kp:DependentProp*, but the cardinality constraint is specific in this class.

kp:CompoundProp

rdfs:subClassOf kp:DependentProp

kp:AndProp
rdfs:subClassOf kp:CompoundProp

kp:OrProp
rdfs:subClassOf kp:CompoundProp

kp:NegProp
rdfs:subClassOf kp:CompoundProp
kp:isDependentOn (primitive) cardinality = 1

Note: property *isDependentOn* has been defined in superclass *kp:DependentProp*, but the cardinality constraint is specific in this class.

KP ontology can used as a markup language to annotate web documents. Since the unit of web information to be processed by KP is proposition, a web document may be annotated as one or more KP propositions. In the next subsection, we introduce how to annotate web documents by an example.

A KP reasoner is implemented in Java to infer the validity and origin of a questioned KP proposition. This reasoner and online demo are also available at *www.eil.utoronto.ca/kp*.

3.7.2 Application Example

In the following, we illustrate how to annotate web information with KP tags and how the axioms are used to determine the provenance of propositions. Rather than maintain provenance in separate metadata documents, KP metadata is embedded directly in web document containing the propositions, making it easier to read and maintain.

An information creator may define a KP proposition as a whole web page, a paragraph, a sentence, even a proper phrase. A KP proposition is annotated by a pair of tags representing one of Asserted_prop, Derived_prop, Equivalent_prop, And_prop, Or_prop,

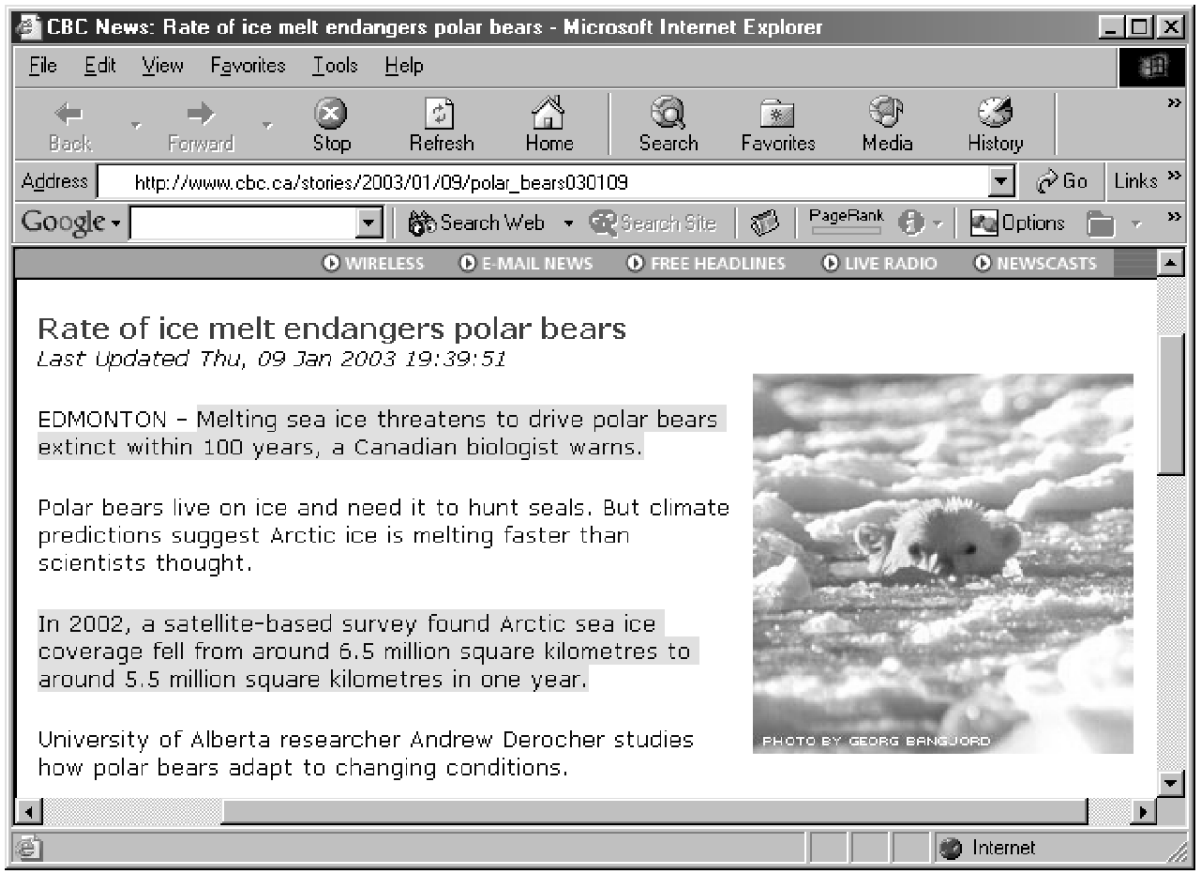

Figure 3.2: Example for Web document annotation

and Neg_prop. For example, <kp:DerivedProp> and </kp:DerivedProp> is used to annotate a Derived_prop. Each KP proposition has certain properties. These properties are given as metadata in the form of either xml attributes or xml elements.

We demonstrate how to make KP annotation by the following example. Assume the web document to be annotated is a html file[3], in which two pieces of text marked with yellow background are two specified KP_props, as shown in the following figure.

Assume that sentence "Melting sea ice threatens to drive polar bears extinct within 100 years" is specified by the information creator as a Derived_prop. This proposition has the following properties.

[3]This example is adapted from a CBC news report.
http://www.cbc.ca/news/story/2003/01/09/polar_bears030109.html

Prop ID: *http://example.com/polar_bears030109.html#EndangeredPolarBears*

prop_content: *Melting sea ice threatens to drive polar bears extinct within 100 years*

author: *Andrew Derocher*

publisher: *CBC*

is_dependent_on: *http://example.com/polar_bears030109.html#MeltingArcticSeaIce*

assigned_truth_value: *True* **in_field**: *Polar Bears*

in_field: *Arctic Environment*

Annotated with KP tags defined in KP Ontology, this proposition appears as follows:

<kp:DerivedProp rdf:about="EndangeredPolarBears"

kp:assignedTruthValue = "True"

kp:isDependentOn = "#MeltingArcticSeaIce"

kp:hasAuthor = "Andrew Derocher"

kp:hasPublisher = "CBC">

<kp:propContent rdf:about="EndangeredPolarBears_content">

Melting sea ice threatens to drive polar bears extinct within 100 years

</kp:propContent>

<kp:inField kp:value = "Polar Bears"/>

<kp:inField kp:value = "Arctic Environment"/>

</kp:DerivedProp>

This Derived_prop is dependent on another KP_prop in the same web document, with text "Arctic sea ice coverage fell from around 6.5 million square kilometres to around 5.5 million square kilometres in 2002." This proposition is an Equivalent_prop equivalent to an Asserted_prop in another web document.

The following is the whole document annotated with KP tags, including the digital signature (in XML-Signature syntax [178]) of the first proposition.

Document1: *http://www.example.com/polar_bears030109.html*

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:dsig = "http://www.w3.org/2000/09/xmldsig#"

xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:kp = "http://www.eil.utoronto.ca/kp#"

>

<body>

< h1 >Rate of ice melt endangers polar bears< /h1 >

<kp:DerivedProp rdf:about="EndangeredPolarBears"

kp:assignedTruthValue = "True"

kp:isDependentOn = "#MeltingArcticSeaIce"

kp:author = "Andrew Derocher"

kp:publisher = "CBC">

<kp:propContent rdf:about="EndangeredPolarBears_content">

Melting sea ice threatens to drive polar bears extinct within 100 years

</kp:propContent>

<kp:inField kp:value = "Polar Bears"/>

<kp:inField kp:value = "Arctic Environment"/>

</kp:DerivedProp>

, a Canadian biologist warns.

< Signature ID="Derocher-polarBears" >

< SignedInfo >

< CanonicalizationMethod

Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>

<SignatureMethod

Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>

<Reference URI="#EndangeredPolarBears">

<DigestMethod

Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>j6hm43k9j3u5903h4775si83...=</DigestValue>

</Reference>

<Reference URI="#EndangeredPolarBears_content">

<DigestMethod

Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>g79lk20rjf023rr032kr93kjr...=</DigestValue>

</Reference>

</SignedInfo>

<SignatureValue>M459ng9784t...</SignatureValue>

<KeyInfo>

<X509Data>

<X509SubjectName>...</X509SubjectName>

<X509Certificate>MIID5jCCA0+gA...lVN</X509Certificate>

</X509Data>

<KeyInfo>

</Signature>

...

< p >A satellite-based survey found

<kp:EquivalentProp rdf:id="MeltingArcticSeaIce"

kp:isDependentOn =

"http://example.org/arcticseaice.html#MeltingArcticSeaIce">

Arctic sea ice coverage fell from around 6.5 million square kilometres to around 5.5 million square kilometres in 2002.

</kp:EquivalentProp>

< /p >

</body>

</html>

Assume that the digital signature of the first proposition is verified successfully, and Andrew Derocher is trusted in the field of Polar Bears by the person requesting knowledge provenance. Therefore, according to theorem KP-1, this *DerivedProp* "EndangeredPolarBears" will be (conditionally) believed. But to determine whether its truth value is to be believed, KP reasoner has to determine whether the proposition it depends on has a believed_truth_value of true. The proposition "MeltingArcticSeaIce" is an equivalent proposition, its truth depends on the truth of its support proposition "http://example.org/arcticseaice.html#MeltingArcticSeaIce" in another web document. The second document is annotated as follows.

Document2: *http://example.org/arcticseaice.html*

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:dsig = "http://www.w3.org/2000/09/xmldsig#"

xmlns:kp = "http://www.eil.utoronto.ca/kp#"

>

<body>...

<kp:AssertedProp rdf:id="MeltingArcticSeaIce"

kp:publisher = "NASA"

kp:inField ="Arctic Environment"

>

<kp:propContent rdf:about="MeltingArcticSeaIce_content">

Arctic sea ice coverage fell from around 6.5 million square kilometres to around 5.5 million square kilometres in one year.

</kp:propContent>

</kp:asserted_prop>

<Signature ID="nasa–meltingArctic">

... similar to dcument1, omitted ... </Signature>

</body>

</html>

In document 2, "MeltingArcticSeaIce" is an *AssertedProp*. From axiom KP-10, by default, it has assigned truth value of true. Its believed truth value depends on whether its information creator is trusted. The publisher of the document is NASA. Assume the digital signature is verified successfully, and the provenance requestor trusts NASA in the field of "Arctic Environment". Then, by theorem KP-1 and 2, "MeltingArcticSeaIce" has a believed_truth_value of true. Consequently, the equivalent proposition "MeltingArctic-SeaIce" in document 1, which has the same content as the one in document 2, also has a believed truth value of true (by axiom KP-6), and finally, the derived proposition "EndangeredPolarBears" has a believed truth value of true.

This example has a online demo at $http://www.eil.utoronto.ca/kp/$.

In our web implementation, a KP reasoner is developed in the form of Java APIs, so that this reasoner can be integrated in various web applications. An online KP reasoned can be used by end-users to get answers for their provenance requests. This KP reasoner can process xml and xhtml types of web documents annotated with KP tags. KP tags are defined in OWL, which follows xml syntax. Inside the KP reasoner, JDOM is employed to extract from web documents the KP_props annotated with KP tags as xml elements. When all provenance information about a KP_prop are extracted, the KP reasoner can infer the believed truth value of this proposition. If a KP_prop is dependent on other KP_props in the same or other web documents, the KP reasoner will crawl the web documents to collect provenance related attributes and infer the origin and believed truth values of those support propositions. Because the focus of this thesis is the logic

of KP, we do not further discuss the technical details about implementation. The web implementation of KP will appear in a paper [92].

The output from the KP reasoner for the example given above can be found in Appendix C.

A bigger application case of KP can be found in Chapter 7.

3.8 Summary and Discussion

Knowledge provenance is an approach to determining origin and validity of knowledge / information on the Web by means of modeling and maintaining information sources, information dependencies and trust relationships. In this chapter, we conceptualized and axiomatized static Knowledge Provenance in the form of ontology. Static KP focuses on static and certain information. Static KP provides the essential concepts of KP and building blocks for dynamic and uncertain KP models.

The information unit considered in KP is a proposition, called KP_prop. A KP_prop can be a sentence, a phrase, a paragraph, an xml element, or a whole web document. Each KP_prop has provenance related properties such as information sources, information dependencies, the assigned truth value given by information creator(s), and a believed truth value evaluated by an information user. In KP, we use FOL to elaborate a 3-valued propositional logic of KP_props.

The proposed static KP ontology can be justified from three aspects: consistency, completeness, and usefulness.

Regarding consistency, the logical consistency (or satisfiability) is easy to be proved, because a knowledge base constructed from the proposed ontology is a set of "general program clauses". Actually, the given application example itself is an interpretation making the ontology satisfiable; the consistency of the ontology's semantics is proved by theorem KP-4.

Regarding completeness, we mean the proposed ontology can answer all competency questions.

The usefulness of the proposed static KP ontology is supported by the application example as well as the application case in Chapter 7.

Chapter 4

Dynamic Knowledge Provenance

This Chapter focuses on dynamic knowledge provenance. In the real world, the validity of information may change over time. Consider the supply chain, the prices of products change over time, inventory changes over time, warehouse space changes over time, etcetera. This chapter introduces *Dynamic Knowledge Provenance* to address the problem of how to determine the validity of information that changes over time.

Similar to static KP, we build dynamic KP ontology in 4 steps: (1) motivating scenario; (2) informal competency questions; (3) terminology; and (4) axioms.

4.1 Motivating Scenario

In the following, the underlying concepts of Dynamic Knowledge Provenance are explored in the following two case studies.

Consider a story in an IT supply chain composed of a reseller (FS), a distributor (DT) and a manufacturer (HP). The reseller (FS) keeps receiving requests from customers about desktop computers configured with 3.06G Pentium 4 processor, so FS sends an asserted proposition, "There is a increasing market demand for desktops with 3.06G Pentium 4 processor", to its major supplier - distributor (DT). The sales department of the distributor (DT) forwards the message to the product management department which

is responsible for product supply. That is, in the terms of KP, the sales department generates an equivalent proposition with the same proposition content as the assertion made by FS. Then, the product management department requests the product information from its major supplier - manufacturer HP. HP replies an asserted proposition, "10,000 desktop PCs configured with 3.06G Pentium 4 processor are available" (effective from 2003-05-26 to 2003-06-01). Based on the asserted proposition made by HP, the equivalent proposition created by the sales department, and some other factors, for example, the financial constraints of the distributor enables it to order 8000 before 2003-05-31, the product management department recommends to head office a product order plan, (actually a derived proposition,) "We should order 8,000 desktop PCs configured with 3.06G Pentium 4 processor from HP" (effective from 2003-05-26 to 2003-05-31).

Case 1: Proposition's truth value is effective within a specified period

In this example, the truth value of the asserted proposition made by HP is effective during period from 2003-05-26 to 2003-06-01. Assume the distributor trusts HP's product supply information, so the asserted proposition is trusted to be true at any time point within the specified period. But after this period, the truth value of the proposition becomes expired, so it will no longer be believed as "true".

Furthermore, the truth value of the derived proposition made by the product management department of DT is effective only during period from 2003-05-26 to 2003-05-31, due to some hidden facts in the derivation. After that period, the proposition become expired and no longer has a truth value. Therefore, the derived proposition is trusted to be true at a given time point, if the time point is within the specified period of the derived proposition and all its dependency propositions are believed to be true at the time point.

Case 2: Information creator is trusted only within a specified period

The information creator may be trusted only within a specified period also. In the example above, assume there is a contract between the distributor (DT) and the reseller

(FS) effective from 2002-04-01 to 2003-12-31. During this period, the distributor trusts the market demand information provided by the reseller to be true. However, if the contract is expired, the distributor will no longer trust the information provided by the reseller anymore. For example, if the contract is effective only from 2002-04-01 to 2003-03-31, the assertion made by the reseller, "There is an increasing market demand for desktops with 3.06G Pentium 4 processor", will not be trusted on 2003-05-26.

These examples reveal some important points for building Dynamic Knowledge Provenance.

- The truth value of an asserted/derived proposition may be effective (existing) only in a specified period;
- Conjunctive propositional dependencies may give rise to a smaller, or null periods of truth value validity;
- Disjunctive propositional dependencies may give rise to discontinuous periods of truth value validity;
- A proposition creator may be trusted in a topic only within a specified period. So, trust relations further constrain the periods of truth value validity.

4.2 Informal Competency Questions

Dynamic Knowledge Provenance needs to answer the following informal competency questions:

- At a given time point, what is the believed truth value of this proposition?
- At a given time point, is this information creator trusted in a field which the proposition belongs to?
- From what time point is this original proposition effective?

- Until what time point is this original proposition effective?
- At a given time point, is this original proposition effective?

4.3 Methodology and Terminology

This section first introduces the time ontology on which dynamic KP is constructed, then defines the terminology for our dynamic KP ontology.

Since many-sorted FOL is used for KP representation, in order to extend static KP to dynamic KP, a new sort is introduced for representing time points.

$\boldsymbol{T}$: the set of time points.

4.3.1 Time Ontology

In Computer Science and Philosophy, many temporal theories have been developed [5] [164] [83], which provide logics of time for developing dynamic KP ontology. The time ontology that we choose for dynamic KP is described as follows.

- Time is a physical dimension [83], which is represented as a continuous line ("having the structure of the real numbers" [164] pp.36). "$<$" is a binary relation, denoting temporal precedence, i.e. "$x < y$" means time point x is earlier than time point y; "x=y" means x is the same time point as y; $x \leq y$ means "$x < y$" or "x=y".
- time point is the primitive temporal object [164], and time interval is defined with time points;
- things that have a temporal extent must have a life-span [83];
- a property has temporal property of homogeneity [5], i.e. if the property holds over an interval, then it holds over all subinterval and points in that interval;

- the logical form of the temporal propositions is "reified" proposition [164](pp.37).

More specifically in KP, every Origin_prop (i.e. Asserted_prop and Derived_prop) has a life-span, called "effective period" in this thesis; believed truth value is homogeneous, and belief is also homogeneous; same as in static KP, KP_props are represented as objects. These objects are manifestations of "reified" propositions. The difference is that a "reified" proposition is a term, and the only property of it concerned is truth value; an object representing a proposition in KP has more concerned properties related to the provenance of that proposition, e.g. assigned truth value, author, the time of creation, and so forth.

Comparing to existing temporal logics, dynamic KP is simpler. Instead of inferring the time interval(s) in which a proposition is true, dynamic KP only needs to answer whether a proposition is believed to be true or false at a given time point.

4.3.2 Effective Periods

As shown in section 1, the truth value of a proposition takes effect in a specified period, in other words, a proposition has a "life-span" as Hayes called [83] (pp.13) . We call this period the proposition's *effective period*, and the proposition is called *effective* in this effective period. A proposition is called "effective" at a given time point, if this time point is within the effective period or life-span of this proposition.

Also, a trust relationship must have a life-span. In this specific period, the trust relationship takes effect. This period is called the effective period of this trust relationship. A trust relationship is called "effective" at a given time point, if the time point is within the effective period of the trust relationship. In order to represent effective periods, we define several predicates in table 4.1.

Several predicates defined in Static KP need to be extended with time, as shown in table 4.2.

Table 4.1: Predicates: time-related

Predicate	**Definition**
$made\_at(x, t)$	$\subseteq \boldsymbol{P} \times \boldsymbol{T}$ KP_prop x is created at time point t. This predicate is only applied to Original_props, i.e. x could be Asserted or Derived KP_prop.
$effective\_from(x, t)$	$\subseteq \boldsymbol{P} \times \boldsymbol{T}$ x is effective from time point t. Here, x could be Asserted or Derived KP_prop.
$effective\_to(x, t)$	$\subseteq \boldsymbol{P} \times \boldsymbol{T}$ x is effective till time point t. Here, x could be Asserted or Derived KP_prop.
$effective\_at(x, t)$	$\subseteq \boldsymbol{P} \times \boldsymbol{T}$ KP_prop x is effective at time point t, or t is within the life-span of x.

Table 4.2: Predicates: temporal-extension

Predicate	**Definition**
$trusted\_in\_during(a, c, f, t_1, t_2)$	$\subseteq \boldsymbol{E} \times \boldsymbol{E} \times \boldsymbol{F} \times \boldsymbol{T} \times \boldsymbol{T}$ Agent a trusts information creator c in knowledge field f from time point t_1 to time point t_2. Here, $[t_1, t_2]$ is called trust relation effective period.
$trusted\_in(a, c, f, t)$	$\subseteq \boldsymbol{E} \times \boldsymbol{E} \times \boldsymbol{F} \times \boldsymbol{T}$ Agent a trusts information creator c in knowledge field f at time point t.
$believed(x, a, t)$	$\subseteq \boldsymbol{P} \times \boldsymbol{E} \times \boldsymbol{T}$ Proposition x is believed by agent a at time point t.

Table 4.3: Symbols Representing Logical Systems

$\mathcal{T}_{KP2}$	denotes the set of axioms and theorems for dynamic KP ontology.
$\mathcal{T}_{KP2,KB}$	$\subset \mathcal{T}_{KP2}$ denotes the set of axioms regarding the constraints on the properties of different types of KP_props in dynamic KP.
$\mathcal{KB}_{KP2,rules}$	$\subset \mathcal{T}_{KP2}$ denotes the set of axioms as rules for provenance reasoning in dynamic KP.
$\mathcal{KB}_{KP2,facts}$	denotes a set of ground predicates representing the properties of KP_props related to answering a provenance request in dynamic KP.

4.3.3 Other Symbols

The other symbols to be used in this chapter is listed in table 4.3.

4.4 Axioms

To define and axiomatize dynamic KP ontology, a set of new axioms related to time need to be introduced; some axioms in static KP ontology need to be extended with time, and some others can be directly inherited from static KP.

4.4.1 Axioms Inherited from Static KP

The following axioms are inherited from static KP.

- Axioms about proposition taxonomy, including: axioms A-1,2 and axiom KP-0.

- Axioms about information sources and authentication, including axioms KP-1,& 2.

- Axioms about property constraints, including axioms KP-11, 12, 13, 14 & 15.

- The Negation as Failure rule (NF-rule).

4.4.2 Relations among Time Points

Following the earlier discussion about time ontology, we represent a set of relations among time points as the following axioms.

Axiom T-1:

$$(x < y) \wedge (y < z) \supset x < z. \tag{4.1}$$

$$(x = y) \wedge (y = z) \supset x = z. \tag{4.2}$$

From this axiom, we further have

$$(x \leq y) \wedge (y \leq z) \supset x \leq z. \tag{4.3}$$

Axiom T-2:

$$\forall x, (-INF < x). \tag{4.4}$$

$$\forall x, (x < +INF). \tag{4.5}$$

4.4.3 Effective Periods

This subsection introduces new axioms about proposition's life-span to KP.

Every KP_prop has a life-span, called *effective period* in KP. The truth value of a proposition exists, called *effective*, only within its effective period.

At a given time point, an asserted_prop is effective if the time point is within its effective period (or life-span).

Axiom DKP-1:

$$type(x, Asserted\_prop)$$
$$\wedge effective\_from(x, t_1) \wedge effective\_to(x, t_2) \wedge t_1 \leq t \wedge t \leq t_2$$
$$\supset effective\_at(x, t). \quad (4.6)$$

At a given time point, a Derived_prop is effective if: (1) the time point is within the effective period of the Derived_prop; and (2) its support proposition is effective at the time point.

Axiom DKP-2:

$$type(x, Derived\_prop)$$
$$\wedge effective\_from(x, t_1) \wedge effective\_to(x, t_2) \wedge t_1 \leq t \wedge t \leq t_2$$
$$\wedge is\_dependent\_on(x, y) \wedge effective\_at(y, t)$$
$$\supset effective\_at(x, t). \quad (4.7)$$

At a given time point, an Equivalent_prop is effective if its support proposition is effective at the time point.

Axiom DKP-3:

$$type(x, Equivalent\_prop)$$
$$\wedge is\_dependent\_on(x, y) \wedge effective\_at(y, t)$$
$$\supset effective\_at(x, t). \quad (4.8)$$

At a given time point, a Neg_prop is effective if its support proposition is effective at the time point.

Axiom DKP-4:

$$type(x, Neg\_prop)$$
$$\wedge is\_dependent\_on(x, y) \wedge effective\_at(y, t)$$
$$\supset effective\_at(x, t). \quad (4.9)$$

At a given time point, an And_prop is effective, if all its support propositions are effective at the time point.

Axiom DKP-5:

$$type(x, And\_prop) \wedge is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge (y_1 \neq y_2) \wedge effective\_at(y_1, t)) \wedge effective\_at(y_2, t)) \supset effective\_at(x, t). \quad (4.10)$$

At a given time point, an Or_prop is effective, if at least one of its support propositions is effective at the time point.

Axiom DKP-6:

$$type(x, Or\_prop) \wedge is\_dependent\_on(x, y) \wedge effective\_at(y, t)) \supset effective\_at(x, t). \quad (4.11)$$

Similarly, a trust relationship also has a life-span. Only within that life-span (or effective period), the trustor (information user) trusts the information creator.

Axiom DKP-7:

$$trusted\_in\_during(a, c, f, t_1, t_2) \wedge t_1 \leq t \wedge t \leq t_2 \supset trusted\_in(a, c, f, t). \quad (4.12)$$

4.4.4 Belief at a Time Point

The following axiom extends theorem KP-1 with time.

An Original_proposition (Asserted_prop or Derived_prop) is believed at a given time point, if: the proposition is effective at the given time point, and the information creator of the proposition is trusted at the time point when the proposition is created.

Axiom DKP-8:

$$type(x, Original\_prop)$$
$$\wedge\, has\_authentic\_creator(x, c) \wedge made\_at(x, t_0) \wedge in\_field(x, f)$$
$$\wedge\, effective\_at(x, t) \wedge trusted\_in(a, c, f, t_0)$$
$$\supset believed(x, a, t). \quad (4.13)$$

4.4.5 Believed Truth Values

In the following, we extend axioms in static KP with time to derive the believed truth value of a KP_prop.

At a given time point, an Asserted_prop has its believed truth value as assigned truth value, if the Asserted_prop is believed by the provenance requester agent at the time point, and the proposition is effective at the time point.

Axiom DKP-9:

$$type(x, Asserted\_prop)$$
$$\wedge\, believed(x, a, t) \wedge assigned\_truth\_value(x, v)$$
$$\supset believed\_truth\_value(a, x, v, t). \quad (4.14)$$

This axiom extends theorem KP-2 with time.

At a given time point, the believed truth value of a Derived_prop is "True" or "False" as its assigned truth value, if the Derived_prop is effective, believed, and its support proposition (premise) is "True".

Axiom DKP-10:

$$type(x, Derived\_prop) \wedge assigned\_truth\_value(x, v) \wedge believed(x, a, t) \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, True, t) \supset believed\_truth\_value(a, x, v, t). \quad (4.15)$$

This axiom extends theorem KP-3 with time.

At a given time point, the believed truth value of an Equivalent_prop is the same as the believed truth value of the proposition it depends on.

Axiom DKP-11:

$$type(x, Equivalent\_prop) \wedge is\_dependent\_on(x, y) \wedge prop\_content(x, c_1) \wedge prop\_content(y, c_2) \wedge equivalent\_to(c_1, c_2) \wedge believed\_truth\_value(a, y, v, t) \supset believed\_truth\_value(a, x, v, t). \quad (4.16)$$

This axiom extends axiom KP-6 with time.

At a given time point, the believed truth value of a negative-proposition is the negation of the believed truth value of the proposition it is dependent on.

Axiom DKP-12:

$$type(x, Neg\_prop) \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, v, t) \supset believed\_truth\_value(a, x, neg(v), t). \quad (4.17)$$

Where, neg(v) is a function to mimic logical negation operator $\neg$, as defined in the previous Chapter. This axiom extends axiom KP-7 with time.

At a given time point, the believed truth value of an And_prop is "True" if all its support propositions are "True" at the time point; The believed truth value of an and-prop is "False" if at least one of support propositions is "False".

Axiom DKP-13a:

$$type(x, And\_prop)$$
$$\wedge\, is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge (y_1 \neq y_2)$$
$$\wedge\, believed\_truth\_value(a, y_1, True, t) \wedge believed\_truth\_value(a, y_2, True, t))$$
$$\supset believed\_truth\_value(a, x, True, t). \quad (4.18)$$

Axiom DKP-13b:

$$type(x, And\_prop)$$
$$\wedge\, is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, False, t)$$
$$\supset believed\_truth\_value(a, x, False, t). \quad (4.19)$$

This axiom extends axiom KP-8 with time.

At a given time point, the believed truth value of an Or_prop is "True" if at least one of its support propositions are "True" at the time point; the believed truth value of an and-prop is "False" if all its support propositions are "False".

Axiom DKP-14a:

$$type(x, Or\_prop)$$
$$\wedge\, is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, True, t)$$
$$\supset believed\_truth\_value(a, x, True, t). \quad (4.20)$$

Axiom DKP-14b:

$$type(x, Or\_prop)$$
$$\wedge\, is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge (y_1 \neq y_2)$$
$$\wedge\, believed\_truth\_value(a, y_1, False, t) \wedge believed\_truth\_value(a, y_2, False, t))$$
$$\supset believed\_truth\_value(a, x, False, t). \quad (4.21)$$

This axiom extends axiom KP-9 with time.

The believed truth value of a KP-proposition at a time point is "Unknown" if and only if either "True" or "False" cannot derived.

Axiom DKP-15:

$$type(x, KP\_prop)$$
$$\wedge\, \neg believed\_truth_value(a, x, True, t) \wedge \neg believed\_truth\_value(a, x, False, t)$$
$$\supset believed\_truth\_value(a, x, Unknown, t). \quad (4.22)$$

This axiom extends axiom KP-11 with time.

Symbol $\mathcal{KB}_{KP2,rules}$ is introduced to denote all rules to be used for provenance reasoning in dynamic KP.

$\mathcal{KB}_{KP2,rules}$ = {axioms A_1, 2, axioms KP_0, 1, 2, axioms T_1, 2, axioms DKP-1, ..., 15}.

4.4.6 Property Constraints

In dynamic KP, similar to static KP, each type of KP_props has property constraints. This subsection discusses the cardinality of temporal properties.

For any Original_prop, the time on which this proposition is created must be uniquely specified; this proposition has only one start time point and has only one end time point of its effective period.

Axiom DKP-16:

$$type(x, Original\_prop) \supset$$
$$((\exists t, made\_at(x,t)) \wedge (made\_at(x,t_1) \wedge made\_at(x,t_2) \supset t_1 = t_2)). \quad (4.23)$$

$$type(x, Original\_prop) \supset$$
$$(effective\_from(x,t_1) \wedge effective\_from(x,t_2) \supset t_1 = t_2). \quad (4.24)$$

$$type(x, Original\_prop) \supset$$
$$(effective\_to(x,t_1) \wedge effective\_to(x,t_2) \supset t_1 = t_2). \quad (4.25)$$

Note that in an implementation, the default effective period of an original proposition is from the time of the creation of a proposition to the positive "infinite" of time-line. In other words, if there is no *effective_from* is specified, the default start time point of the effective period is the the time point of the creation of this proposition; if there is no *effective_to* is specified, *+INF*, this proposition is supposed to be effective till the positive infinite of the time-line.

Let $\mathcal{T}_{KP2,KB}$ denote the set of axioms regarding property constraints or cardinality in dynamic KP, that is,

$\mathcal{T}_{KP2,KB}$ = {Axioms KP_11, 12, 13, 14 & 15, Axiom DKP-16}.

In Dynamic KP, $\mathcal{KB}_{KP2,facts}$ denotes the set of grounded predicates that define all KP_prop instances and their primary properties (defined properties). $\mathcal{KB}_{KP2,facts}$ contains a finite number of ground atomic formulas.

Similar to static KP, $\mathcal{KB}_{KP2,facts}$ must satisfy $\mathcal{T}_{KP2,KB}$. This constraint leads to the following axiom.

Axiom DKP-17:

$$\mathcal{KB}_{KP2,facts} \supset \mathcal{T}_{KP2,KB} \quad (4.26)$$

Now, let

$$\mathcal{T}_{KP2,KB}{}^{+} = \mathcal{T}_{KP2,KB} \cup \{AxiomDKP\_17\}.$$

By this axiom, the axioms in $\mathcal{T}_{KP2,KB}$ regarding cardinality define the structure of the instance definition of each type of KP_props.

4.5 Reasoning with Negation

Similar to static KP, dynamic KP also use Negation as Failure rule to infer negative literals.

Let $\mathcal{KB}_{KP2}$ denote a knowledge base constructed from dynamic KP ontology, i.e.

$$\mathcal{KB}_{KP2} = \mathcal{KB}_{KP2,rules} \cup \mathcal{KB}_{KP2,facts}.$$

$\mathcal{KB}_{KP2,rules}$, consists of only Horn clauses and one "general program clause" (axiom DKP-11); $\mathcal{KB}_{KP2,facts}$ contains only atoms, a specific form of Horn clauses. So, NF-rule can be applied in $\mathcal{KB}_{KP2}$.

This solution is based on the following knowledge completeness assumption.

Knowledge Completeness Assumption for Dynamic KP (KCA-KP2): $\mathcal{KB}_{KP2,facts}$ and $\mathcal{KB}_{KP2,rules}$ are all the knowledge used to infer the provenance properties of a *KP_prop* defined in $\mathcal{KB}_{KP2,facts}$.

The Negation as Failure rule (NF-rule) remains the same as stated in static KP.

The set of axioms for dynamic KP ontology using Negation as Failure can be represented as

$$\mathcal{T}_{KP2} = \mathcal{KB}_{KP2,rules} \cup \mathcal{T}_{KP2,KB}{}^{+} \cup \{KCA_{KP2}, NF\_rule\}.$$

4.6 Consistency and Completeness

In order to justify the proposed dynamic KP ontology, similar to what we did in static KP, we need to discuss the consistency and completeness of dynamic KP ontology.

4.6.1 Consistency

Since $\mathcal{KB}_{KP2}$ is a set of general program clauses, it is easy to prove the logic consistency of $\mathcal{T}_{KP2}$. In the following, we discuss the consistency of the semantics of the proposed dynamic KP ontology.

Same as in static KP, if a KP_prop has two (or more) different (either assigned or believed) truth values, the semantics becomes inconsistent; on the other hand, if one and only one truth value of a KP_prop is assigned and can be derived, then the semantics of the ontology is consistent.

Axiom KP-12 guarantees that each Original_prop has one and only one assigned truth value of either "True" or "False".

Theorem KP-4 for static KP, regarding the unique believed truth value of a KP_prop, needs to be extended with time as follows.

Theorem DKP-1: *By system $\mathcal{T}_{KP2}$, given $\mathcal{KB}_{KP2,facts}$ that satisfies $\mathcal{T}_{KP2,KB}$, for any provenance requester a, at any time point t, any KP_prop x has one and only one believed truth value of either "True", "False" or "Unknown".*

$$
\begin{aligned}
&\mathcal{KB}_{KP2,rules} \models \\
&\qquad \mathcal{KB}_{KP2,facts} \supset \\
&\qquad (type(x, KP\_prop) \supset \\
&\qquad ((believed\_truth\_value(a, x, True, t) \vee believed\_truth\_value(a, x, False, t) \\
&\qquad \vee believed\_truth\_value(a, x, Unknown, t)) \\
&\qquad \wedge \neg(believed\_truth\_value(a, x, True, t) \wedge believed\_truth\_value(a, x, False, t)) \\
&\qquad \wedge \neg(believed\_truth\_value(a, x, True, t) \wedge believed\_truth\_value(x, Unknown, t)) \\
&\qquad \wedge \neg(believed\_truth\_value(a, x, False, t) \wedge believed\_truth\_value(a, x, Unknown, t))))
\end{aligned}
\tag{4.27}
$$

The proof of this theorem is very similar to the proof of theorem KP-4, except that whether the given time point is within a life-span needs to be considered, in order to derive a believed truth value (i.e. "True" and "False"). The detail of proof is omitted.

4.6.2 Completeness

We discuss the completeness of the ontology in the sense of that the ontology can answer every competency question.

Given the provenance requester R, the concerned time point T, and questioned KP_prop X, the competency questions presented earlier is briefly discussed one by one as follows.

Q1. At a given time point, what is the believed truth value of this proposition?

This is the major question that the dynamic KP ontology needs to answer. Given provenance requester R, time point T, and questioned KP_prop X, this question can be formally represented as follows.

$$type(X, KP\_prop) \supset \exists v, believed\_truth\_value(R, X, v, T). \tag{4.28}$$

This question can be answered by theorem DKP-1.

Q2. At a given time point, is this information creator trusted in a field which the proposition belongs to?

This question can be formally represented as follows.

$$type(X, Original\_prop) \wedge$$
$$(\exists c, f, has\_infoCreator(X, c) \wedge in\_field(X, f) \wedge trusted\_in(R, c, f, T)). \tag{4.29}$$

By axiom KP-12,

$$\exists c, f, has\_infoCreator(X, c) \wedge in\_field(X, f)$$

must be true. The answer to the question is either True or False, dependent on the type of X and the return of external predicate

$$trusted\_in(R, c, f, T).$$

Q3. From what time point is this original proposition effective?

This question can be formally represented as follows.

$$type(X, Original\_prop) \supset (\exists t, effective\_from(X, t)). \tag{4.30}$$

By axiom DKP-12, the answer to this question must be True and returns a value of t.

Q4. Until what time point is this original proposition effective?

This question can be formally represented as follows.

$$type(X, Original\_prop) \supset (\exists t, effective\_to(X, t)). \tag{4.31}$$

By axiom DKP-12, the answer to this question must be True and returns a value of t.

Q5. At a given time point, is this original proposition effective?

This question can be formally represented as follows.

$$type(X, Original\_prop) \wedge effective\_at(X, T). \tag{4.32}$$

By axiom DKP-1 & 2, the answer to this question is either True or False.

In summary, this dynamic KP ontology can answer all predefined competency questions.

4.7 Temporal Extension of Web Ontology

For dynamic KP, the web ontology described in the previous chapter needs to be extended with time. A small set of new properties related to time are listed as follows.

kp:KPProp
kp:effectiveAt (derived) cardinality = 1

kp:OriginalProp
kp:madeAt (primitive) cardinality = 1
kp:effectiveFrom (primitive) cardinality = 1
kp:effectiveTo (primitive) cardinality = 1

In the web ontology, we use xml schema datatype *xsd:dateTime* [1] to represent time.

4.8 Example

The following example from our motivating scenario illustrates KP annotation in web documents and knowledge provenance reasoning.

Consider the example discussed in section 3 regarding an IT supply chain composed of a reseller (FS), a distributor (DT) and a manufacturer (HP). As shown in figure 1, the product management department of the distributor (DT) created a derived proposition, "We should order 8,000 desktop PCs configured with 3.06G Pentium 4 processor from HP" (effective from 2003-05-26 to 2003-05-31), which is dependent on two propositions: (1) an equivalent proposition created by the sales department of the distributor stating "There is an increasing market demand for desktops with 3.06G Pentium 4 processor", which is dependent on an asserted proposition with same proposition content created by a contracted reseller called FS; and (2) an assertion created by manufacturer HP that says that "10,000 desktop PCs configured with 3.06G Pentium 4 processor are available" (effective from 2003-05-26 to 2003-06-01).

[1] refer to: http://www.w3.org/TR/xmlschema-2/#dateTime

4.8.1 KP Annotation

The web document that contains the derived proposition and its dependency And_prop created by the product management department can be embedded with KP metadata as follows. The annotation to other web documents is in a similar way.

Document: http://www.pm.examp.com/doc4

<HTML xmlns="http://www.w3.org/1999/xhtml"

xmlns:dsig = "http://www.w3.org/2000/09/xmldsig#"

xmlns:kp = "http://www.eil.utoronto.ca/kp#"

>

<body>

...

<kp:DerivedProp rdf:id= "order_PCP4"

kp:assignedTruthValue = "True"

kp:isDependentOn ="#demand_supply_PCP4"

kp:author = "Product Management Department"

kp:inField = "Supply"

kp:effectiveFrom = "2003-05-26"

kp:effectiveTo = "2003-05-31"

>

<kp:propContent>

We should order 8,000 desktops configured with 3.06G Pentium 4 processor from HP

</kp:propContent>

</kp:DerivedProp>

<kp:AndProp rdf:id= "demand_supply_PCP4"

kp:isDependentOn = "http://www.hp.examp.com/doc2#available_PCP4_HP"

<kp:isDependentOn = "http://www.hp.examp.com/doc3#demands_PCP4"

>

<Signature ID= "ProdMgmt–order-PCP4">

<SignedInfo>

<CanonicalizationMethod Algorithm= "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>

<SignatureMethod

Algorithm= "http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>

<Reference URI= "#order_PCP4">

<DigestMethod Algorithm= "http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>j6hm43k9j3u5903h4775si83... </DigestValue>

</Reference>

</SignedInfo>

<SignatureValue>M459ng9784t...</SignatureValue>

<KeyInfo>

<X509Data>

<X509SubjectName>...</X509SubjectName>

<X509Certificate>MIID5jCCA0+gA...lVN

</X509Certificate>

</X509Data>

<KeyInfo>

</Signature>

</body>

</html>

4.8.2 KP Reasoning

In this example, assume the trust relation effective periods (associated with contracts) are as follows: HP is trusted by DT from 2002-01-01 to 2004-12-31; FS is trusted by DT from 2002-04-01 to 2003-12-31; and in case (c) of figure 1, FS is trusted from 2002-04-

01 to 2003-03-31. In addition, assume the sales department and product management department are trusted within the distributor (DT) on topic "Order", "Demands" and "Products", "Supply" respectively. Finally, assume all atomic propositions are made at the time points of "effective_from".

As shown in figure 4.1, case (a) requests the believed truth value of the derived proposition that "We should order 8,000 desktop PCs configured with 3.06G Pentium 4 processor from HP" at time point 2003-05-28. "True" is obtained by reasoning using KP axioms; case (b) requests the believed truth value of the derived proposition at 2003-05-23. "Unknown" is obtained, for the reason that 2003-05-23 is not covered by [2003-05-26, 2003-05-31], the effective period of HP's asserted proposition, which causes the asserted proposition and further the derived proposition are not effective; case (c) requests the believed truth value of the derived proposition at 2003-05-28. "Unknown" is reached because 2003-05-28 is not covered by [2002-04-01,2003-03-31], the effective period of trust to reseller FS, which causes that FS cannot be trusted and further its assertion and all proposition dependent on it cannot be trusted also.

This example demonstrated how KP is embedded in information flow and how KP reasoning is conducted.

4.9 Summary

In this chapter, we extend static KP ontology, the core formal model of KP constructed in the previous chapter, into dynamic KP ontology, to address the provenance problem in the world where the truth of propositions and the trust relationships change with time.

In building dynamic KP, time is regarded as a continuous line; time points are the primary temporal objects, on which time intervals are constructed; KP_props (and trust relationships) have their life-spans; the properties of KP_props have temporal property of homogeneity.

Dynamic KP adopts a simple solution. First, *effective periods* are introduced for KP_props and trust relationships, to represent their life-spans; then, to infer the believed truth value of a proposition at a given time point, whether this proposition as well as its support propositions is effective at this time point, needs to be considered.

This dynamic KP ontology can answer the questions that at a given time point, what is the believed truth value of a KP_prop; but it cannot answer the questions that in which time periods, the believed truth value of a KP_prop is true/false/unknown. How to efficiently answer the later type of questions still remains an open issue.

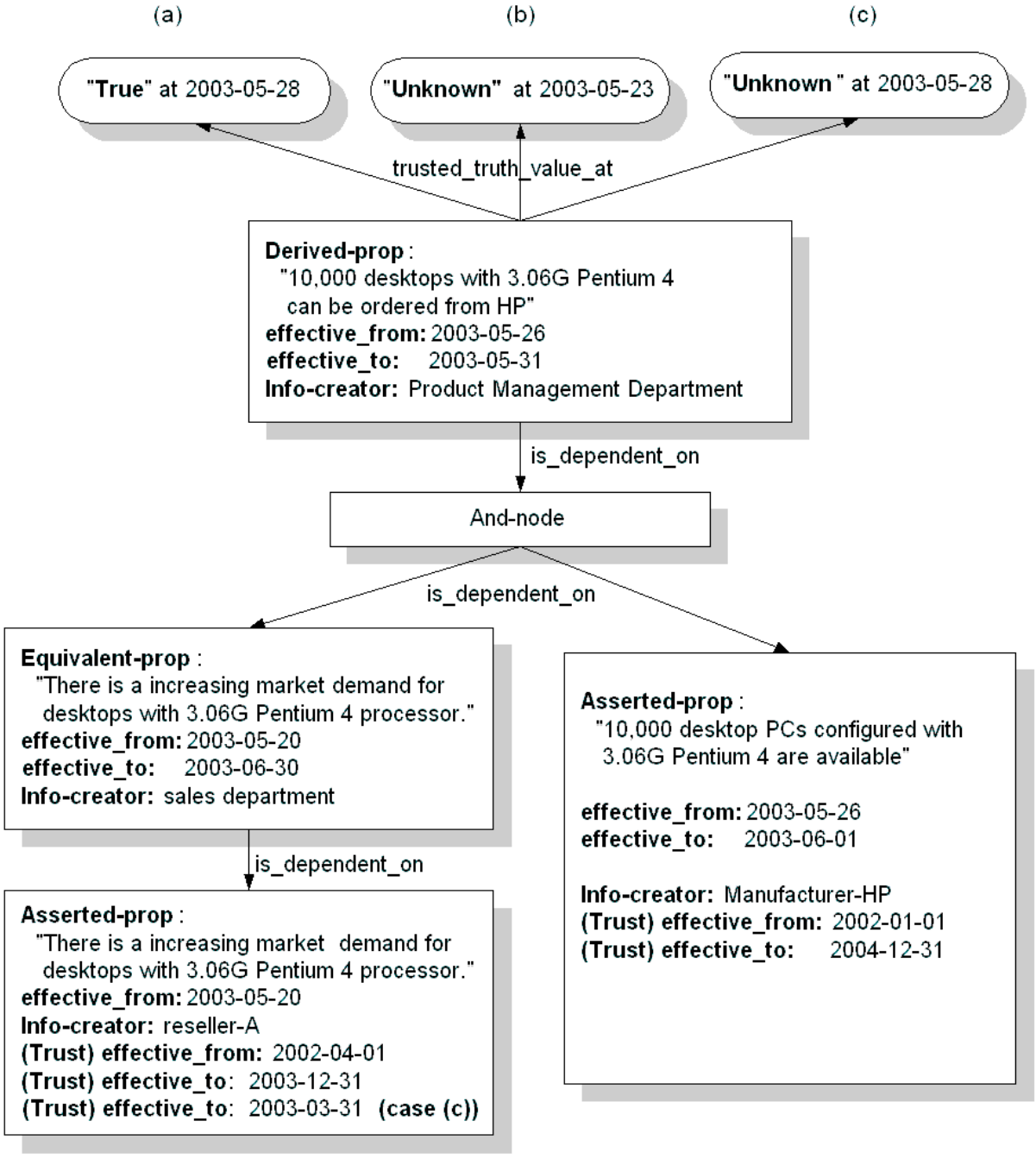

Figure 4.1: Application example of dynamic Knowledge Provenance

Chapter 5

A Logic Theory of Trust

Knowledge provenance (KP) is an approach used to determine the origin and validity of web information, by means of modeling and maintaining information sources, information dependencies, and trust structures. Basically, in KP, the validity of information is mainly determined with the trust placed on the information sources. In Chapters 3 and 4, we studied static KP and dynamic KP. These models assume that trust relationships between information consumers and information creators have already been calculated, and these models mainly focused on representing information sources and information dependencies and deducing the provenance of information. This chapter and the next two chapters focus on modeling trust structures.

In this chapter, we develop a logical theory of trust that formally presents the semantics of trust, the transitivity of trust, and the conditions for trust propagation in social networks; based upon this logical theory, in chapter 6, we will construct a distributed trust reasoning model, in which personal trust relationships needn't to be published, so that privacy can be better protected in trust reasoning using Web-based social networks.

The contents of this chapter are organized as follows: section 1 introduces why trust is important in computing, and identifies the problems to be solved in trust modeling; section 2 explores the meanings and properties of trust, which provides a conceptual

foundation for formalizing trust; section 3 provides the motivating scenarios for our trust modeling; section 4 gives the methodology and terminology to be used; then in section 5, in the languages of the *situation calculus* and *epistemic logic*, we define the formal semantics of trust and construct a model of trust in situation calculus; section 6 discusses sources of trust; in section 7 and 8, we study the transitivity of trust and trust propagation in social networks; section 9 provides two examples to demonstrate the application of the proposed logical theory of trust; finally, we give a summary on this chapter.

5.1 Introduction

Our specific interest in trust comes from the need to determine the validity of information. As presented in previous chapters, basically, we determine the validity (i.e. believed truth value) of a proposition by considering whether the information user trusts in the information source(s) in a knowledge field which the proposition belongs to.

In general, trust is a widely concerned problem. In the remainder of this section, from a broader view, we discuss why trust is important and the problems exist in the research of trust modeling.

As well known, the Web has become an open dynamic and decentralized information/knowledge repository, a distributed computing platform and a global electronic market. In such a cyberspace, people, organizations and software agents have to interact with "strangers" and have to be confronted with various information with unfamiliar sources. This makes trust arise as a crucial factor on the web.

In general, trust is "a basic fact of social life" [118]. For example, we trust the data that we use for a decision to be reliable; when we cross an intersection of streets, we trust the cars in other directions to follow the traffic signals. Trust exists so widely in our social life that we may even not fully be aware of using it.

Trust is important to us because it is one of the foundations for people's decisions. Generally, rational decisions made in the real world are based on the mixture of bounded rational calculation and trust. According to Simon[169], a decision making process in the real world is limited by "bounded rationality" i.e. the "rational choice that takes into account the cognitive limitations of the decision maker - limitations of both knowledge and computational capacity" . In a real decision situation, since we only have limited information/knowledge and computing capacity as well as limited time available for decision analysis, a rational decision has to be partly based on bounded rational calculation and partly based on trust. As Luhmann revealed, trust functions as "reduction of complexity" in our social life [118].

Returning to the context of trust on the Web, the interest in addressing the issue of trust on the web has appeared under the umbrella of the "Web of Trust". An important issue is how to transplant the trust mechanism in our real world into the cyberspace. To this end, many researchers attempt to construct formal models of trust and to develop tools that support people making trust judgments on the web. Basically trust is established in interaction between two entities. Many models proposed focus on how to calculate and revise trust degrees in interaction between two entities. However, each entity only has a finite number of direct trust relationships, which cannot meet the needs of various interaction with unknown or unfamiliar entities on the Web. As a promising remedy to this problem, social network based trust, in which A trusts B, B trusts C, thus A indirectly trusts C, is receiving considerable attention. A necessary condition of trust propagation in social networks is that trust needs to be transitive. In other words, without transitivity, trust cannot propagate in networks. However, is trust transitive? What types of trust are transitive and why? Few theories and models found so far answer these questions in a formal manner. A major problem in trust modeling is the lack of a clearly defined formal semantics of trust. For this reason, most models mainly focus on how to calculate trust degrees when trust propagates in social networks, and they

either directly assume trust transitive or do not give a formal discussion of why trust is transitive.

To fill this gap, this chapter aims to construct a logical theory of trust. This theory should provide a formal and explicit specification for the semantics of trust; from this formal semantics, this theory should be able to identify the conditions for the transitivity of trust and the conditions for trust propagation in social networks.

5.2 What is Trust?

5.2.1 Meaning of Trust

The Oxford Dictionary of Current English (4th edition, 2006) defines trust as *"firm belief in the reliability, truth, ability of someone or something."* The dictionary further explains trust in information as: *"acceptance of the truth of a statement without proof."* These definitions reflect people's common understanding of trust and the most important characteristics of trust, but they are not sufficient for the purpose of trust formalization.

Synthesizing the concepts of trust discussed in Chapter 2, we have the following view of trust.

Trust is the psychological state comprising (1) ***expectancy****: the trustor expects a specific behavior of the trustee such as providing valid information or effectively performing cooperative actions; (2)* ***belief****: the trustor believes that the expected thing is true, based on evidence of the trustee's competence and goodwill; (3)* ***willingness to be vulnerable****: the trustor is willing to be vulnerable to that belief in a specific context, where the specific behavior of the trustee is expected.*

In trust, there are two roles involved: a trustor and a trustee. Furthermore, there are three aspects in the implications of trust. First, when it is said that entity A trusts B, people must ask a question "A trusts B on what?" This leads to the first implication of trust: expectancy, i.e. the trustor expects that trustee behaves in a specific manner

within a specific context. The expected behavior can be: (1) valid information created by the trustee; or (2) a successful cooperative action conducted by the trustee. Secondly, when trusting, the trustor must believe that the trustee behaves as expected, according to the competence and goodwill of the trustee. This is the most recognized aspect of the meaning of "trust". Thirdly, the trustor not only believes the trustee will behave as expected but also is willing to be vulnerable for that belief in a specific context, i.e. the trustor is willing to assume the risk that the trustee may not behave as expected.

As many researchers realized, trust is context-specific, for example, a person may trust her or his financial advisor about investment analysis but doesn't trust the advisor in health-care. There are two types of contexts related to trust. The first is the context where the expected valid information is created, and the second is the context in which the trusted information (or action) will be applied by trustor or the situation where the trustor is confronted with the trust judgment problem. These two contexts may not be the same. For example, a financial expert (the trustee) created a piece of information in a context of giving financial investment seminar, and in another context of buying stocks, an investor (the trustor) attempts to use this information and needs to judge the validity of it.

In summary, "trust" has three aspects of meaning: (1) expectancy: the trustor expects that trustee behaves in a specific manner in a specific context; (2) belief that the expectancy is true; (3) willingness to be vulnerable for that belief in a specific context. Finally, the meaning of trust we adopt can be expressed as follows:

$$\begin{aligned} Trust = & \, Expectancy \\ & + Belief \ in \ expectancy \\ & + Willingness \ to \ be \ vulnerable \ for \ that \ belief. \end{aligned}$$

5.2.2 Properties of trust

We discuss whether trust has the following properties.

Transitivity

Many researchers have the opinion that trust is generally intransitive. Perhaps for this reason, there is few research regarding trust propagation in social networks in traditional trust studies. However, it is a basic fact that some of trust is transitive, for example, recommendation is a typical type of transitive trust. Transitivity of trust is dependent on what is trusted in. "Trust in belief" (to be discussed later) is transitive. For example, a patient trusts what his family doctor believes in health-care; the doctor trusts the knowledge of heart disease produced by an expert in the area; then the patient indirectly trusts the knowledge. Transitivity is an important foundation for studying social networks-based trust.

Symmetry

Symmetry concerns whether individual A trusts individual B when B trusts A. Trust is generally not symmetrical. For example, a patient trusts a physician on health-care issue, but the physician may not trust the patient on the same issue. However, "trust in goodwill" and "emotional trust" tend to be symmetrical after a relatively long term of dynamic adjustments in the interaction between the trustor and the trustee.

Trust is not necessary to be asymmetrical or anti-symetrical.

Reflexivity

Some people may tend to think that trust is reflexivity, i.e. an individual always trust himself/herself. From our point of view, the following types of trust are reflexive. (Note: the categories are overlapped for they are classes in different views.)

(1) Trust in belief. People always trusts what they believe;

(2) Trust in performance in a context where the trustor has confidence on himself/herself, e.g. an seasoned investor highly trust himself/herself on investment;

(3) Trust in goodwill. In normal situations, people trust themselves to have goodwill toward themselves;

(4) Emotional trust. In normal situations, people have high emotionality toward themselves.

On the other hand, trust in performance in the context where the trustor has no confidence is irreflexive, that is, an individual always does not trust himself/herself in the contexts where he or she has no confidence. For example, a computer user without knowledge of operating system (without trust in himself/herself) does not trust himself/herself to change system parameters.

5.3 Motivating Scenarios

In order to construct a formal model of trust in social networks, we start from motivating scenarios.

Consider the following use cases of the electronic business of a gift company, F. F originally is a floral company. Now its online business includes both bouquets and other gift products. The web services technology makes F able to extend its online store to a virtual omni-gift store that sells not only the products in its catalogues but also any other types of gift products available via web services.

Case 1: Trust in Business Partnerships

James, an old customer of F, wants to order from F's online store a gift for a friend, a special high quality fine china tea set made in Jingdezhen of China. Assume this product is not found in F's catalogues. In F's online store, a sales agent, a software robot representing F, first finds from web services registries (e.g. UDDI registries) a list of service providers who sell the requested products; second, from this list, as a general business rule, the agent needs to check out service providers whose products are trusted

by F to have high quality; then, the agent presents to James a selected list of products provided by the trusted service providers; after James chooses a product, the agent orders the product for him. Here, we ignore the detail of web service process and only focus on how the agent makes trust judgments.

A service provider which F found in a web services registry is J, a porcelain company in Jingdezhen city. In order to maintain F's business reputation, before F recommends J's fine china tea sets to customers, F needs to make sure J's products are of high quality. To make this judgment, the sales agent searches J in F's business relationship management system. Unfortunately, J is not in the system, that is to say, F does not know J at the time. However, F has a long term business partner P. P frequently provides for F porcelain products such as vases. Therefore, F trusts P's judgments on porcelain product quality. Furthermore, P has a major trusted porcelain product supplier S in New York, so similarly P trusts S's judgment on porcelain product quality. Finally J is one of S's trusted suppliers. In this latter relationship, S trusts the product quality of J. From this chain of trust, the sales agent of F infers that the porcelain tea sets of J have high quality.

In the following, we analyze the roles and relationships demonstrated in this case.

James, a customer of F who trusts F's services, wants to buy from F a special high quality fine china tea set. This request activates web service discovery, trust judgments, and trading.

F, a gift company, trusts P on P's judgement on porcelain product quality because P is a long term professional business partner of F. In the terms of trust, the expectancy of F to P is that P's judgement on porcelain product quality is correct, and F feel comfortable to believe what P believes in porcelain product quality.

P, a porcelain product company, trusts S's judgment on porcelain product quality similarly as F does.

S, a porcelain product supplier, trusts J on the quality of the porcelain products J produces. In the terms of trust, S expects that the porcelain products of J have high

quality; based on J's performance in the past, S would like to believe this expectancy.

J, a porcelain manufacturer at Jingdezhen, is unknown to F.

The chain of trust can be summarized as follows: (1) F trusts P's judgement on porcelain product quality; (2) P trusts S's judgement on porcelain product quality; (3) S trusts that the porcelain products of J has high quality; (4) so by (2) and (3), P also trusts that the porcelain products of J has high quality; (5) similarly, by (1) and (4) F trusts that the porcelain products of J have high quality.

Case 2: Trust in System

Consider another situation in which F does not know J, and F also does not find any useful relationship in its business relationship management system to connect to J. However, J shows that it has ISO 9000 quality certification. Assume that F trusts ISO 9000 system thus F trusts J's products meeting ISO 9000 quality standards. This makes F to trust that the porcelain products of J has high quality.

Case 3: Trust in Business Partnerships (2)

Consider the business relationships in case 1. Assume that S trusts P that P is able to and will pay for all its purchases so allows P to pay quarterly rather than to pay in every transaction, and P trusts F in the same way. Now, consider the following situation. For the first time, F orders a considerable amount of porcelain vases directly from S. Does S trusts F on later payment as trusts P? The common knowledge tell us that the answer is "no". This is because the facts that S trusts P to pay later and P trusts F to pay later does not necessarily imply S trusts F to pay later. In other words, this trust is not transitive.

Findings:

From these cases, a number of important facts and concepts can be revealed.

The Factor of Trust in Web Services Based B2B

In e-business based on web services technology, a web services user (here, F) accepts only trusted service providers retrieved from web services registries. The trust issues may arise in many aspects of e-business including service quality, business credibility, as well as business information security and privacy.

Types of Trust

There are different types of trust.

In case 1, the trust which F places on P is the trust in what P believes. The trust P places on S is the same type. This type of trust is called ***trust in belief***. The trust which P places on F in case 3 is the trust in F's performance. This is another type of trust: ***trust in performance***. The other examples of this type of trust include the trust which S places on P in case 3 and S places on J in case 1.

Transitivity of Trust

In case 1, S's trust in J propagates to P and then to F because F trusts in P's belief and P trusts in S's belief on product quality. This fact leads us to a hypothesis: trust in belief is transitive. Because of this property, trust can propagate in business relationship formed social networks.

However, trust is not transitive in general. Case 3 shows that trust in performance is not transitive.

Sources of Trust

Basically, trust comes from the experience of interaction between two parties. For example, F trusts in P's belief on porcelain product quality because P is F's long term business partner and F gradually knows that P is very professional on porcelain business from the business interaction between them. The experience of interaction is the essential source of trust. The trust from interaction experience is called ***inter-individual trust***, and also called ***direct trust***.

Interestingly, from case 1, we have observed the fact that trust may propagate in social networks. Because of trust propagation, F trusts in J's products. This leads to a new type of trust, whose source is from trust propagation in social networks. In this thesis, this type of trust is called ***relational trust*** or ***social networks based trust***.

Finally, in case 2, F trusts in J's products because F trusts in ISO 9000 quality management system and J complies with the system. This is an example of ***system trust***.

5.4 Methodology, Terminology and Competency Questions

Following the ontology development methodology of Gruninger & Fox [74], we specify trust ontology in 5 steps: (i) provide motivating scenarios, which has been given in the last section; (ii) define informal competency questions for which the ontology must be able to derive answers; (iii) define the terminology; (iv) formalize competency questions; (v) develop the axioms (i.e. semantics) of the ontology to answer these questions.

In the following, we first introduce the specific methods in our formalization of trust in the situation calculus; then we define informal competency questions, terminology, and formal competency questions.

5.4.1 Methodology

This chapter aims to construct a logical theory of trust in the form of an ontology. To this end, trust is formalized by using *epistemic logic* and *situation calculus*. We choose epistemic logic and the situation calculus as the tools for formalizing trust for the following reasons. First, epistemic logic has a formal representation of belief, and belief is the kernel element of trust; secondly, situations in the situation calculus provide a solution to formally represent the contexts of trust; finally, trust dynamically changes with the growth of the knowledge related to trust.

In this thesis, we treat trust as relation between a trustor, a trustee, an expected thing, and a context.

Different from *epistemic logic*, in which belief is represented as a modal operator, we adopt a First Order Logic approach. Particularly, in our trust ontology, trust and belief will be represented as fluents in situation calculus. We represent fluents in reified form [147]. That is to say, the fact that a relational fluent is true in a situation is represented by $holds(f(x), s)$ rather than predicate $f(x, s)$. In this way, a fluent is a term. So that a fluent may have other fluents as parameters.

Following Pinto [147], we treat the "logical operators" between fluents as functions. In other words, the "propositional logical expressions" of fluents are treated as functions rather than logical expressions in the language of the situation calculus. A logical functor used for fluents is denoted as the corresponding logical operator with a dot above.

Definition TR-1 (by Pinto [147]):

$$holds(f_1 \dot{\wedge} f_2, s) \equiv holds(f_1, s) \wedge holds(f_2, s) \tag{5.1}$$

$$holds(\dot{\neg} f, s) \equiv \neg holds(f, s) \tag{5.2}$$

Based on the above definitions, other "logical operators" are also defined as functions of fluents. For example,

$$holds(f_1 \dot{\supset} f_2, s) \equiv holds(f_1, s) \supset holds(f_2, s). \tag{5.3}$$

Regarding context representation, McCarthy [126] introduces operator *ist(c,p)*, which is true if proposition p is true in context c. This operator is analogous to $c \supset p$. "A context is modelled by a set of truth assignments." [24] Following this, in our formalization of trust, a context of trust is regarded as a condition of trust. In the language of situation calculus, we will not introduce any new representation, instead we directly use the existing "*fluents*" associated with "*situations*" to represent contexts. In this way, we decide whether it is in a specific context by checking whether the fluent representing the context holds.

The logical theory of trust mainly focuses on the logical relations among trust related fluents. The relations among fluents are called "state constraints" in the language of situation calculus, and several methods have been developed to solve the so called "ramification problem" [147]. In this thesis, in order to focus on the logic of trust and to make the theory easy to be understood, we keep the relation among fluents in the form of state constraints.

In this chapter, we only consider the case of certainty, so believing will certainly lead to willing to be vulnerable. For this reason, we will not explicitly include "willing to be vulnerable" in our formalization. In the cases of uncertainty in which the degree of belief is considered, *willingness to be vulnerable for that belief* is corresponding to a decision to trust or not. This will be discussed in the next chapter.

5.4.2 Informal Competency Questions

In the context of making trust judgments in social networks, the trust ontology under development needs to answer the following competency questions:

Q1: In a specific context, can an entity (trustor) trusts another entity (trustee) regarding what the trustee performs, particularly, the validity of the information created by trustee?

Q2: In a specific context, can an entity (trustor) trusts another entity (trustee)

regarding what the trustee believes?

Q3: When an entity has trust in another entity's belief in a given context, and the second entity has trust in a third entity's performance (or belief) in another context, can the first entity have trust in the third entity's performance (or belief)? if so, in what context?

Q4: When an entity has trust in another entity's performance (or belief) in a specific context, given the information created by the second entity within the context, can the first entity believe this information?

After we define the terminology of the ontology under development, these informal competency questions will be formally represented, then axioms will be developed to answer these competency questions.

5.4.3 Terminology

The situation calculus is a many-sorted logic language. We will use the following sorts:

A: the set of actions;

S: the set of situations;

F: the set of fluents;

E: the set of entities;

Y: the set of activities;

D: the set of domain objects.

In addition, we follow the convention that all unbound variables are universally quantified in the largest scope.

We define the relational fluents and predicates to be used in our trust ontology in Tables 5.1 and 5.2 respectively.

Detailed explanation about the terminology will be given in the context of discussion later. Trust related actions will introduced in Chapter 6. In this chapter we focus on the semantics of trust and its transitivity, which are described in the form of state constraints

Table 5.1: Relational Fluents

Fluent	**Definition**
$believe(d, x)$	$\subseteq \boldsymbol{E} \times \boldsymbol{F}$ Entity d believes that thing x is true. x is a fluent to represent the expectancy in a trust.
$trust\_p(d, e, x, k)$	$\subseteq \mathbf{E} \times \mathbf{E} \times \mathbf{F} \times \mathbf{F}$ denotes trust in performance relationship. In context k, trustor d trusts trustee e on thing x made by e. x is the expectancy of the trust, i.e. the performance or information made by trustee and expected by trustor.
$trust\_b(d, e, x, k)$	$\subseteq \mathbf{E} \times \mathbf{E} \times \mathbf{F} \times \mathbf{F}$ denotes trust in belief relationship. In context k, trustor d trusts trustee e on thing x that e believes.
$has\_p\_tr(d, e, x, k)$	$\subseteq \mathbf{E} \times \mathbf{E} \times \mathbf{F} \times \mathbf{F}$ Trustor d has *trust in performance* type of inter-individual trust relationship with trustee e. k is the context of trust, and x is the expectancy of trust.
$has\_b\_tr(d, e, x, k)$	$\subseteq \mathbf{E} \times \mathbf{E} \times \mathbf{F} \times \mathbf{F}$ Trustor d has *trust in belief* type of inter-individual trust relationship with trustee e.
$made(x, d, k)$	$\subseteq \mathbf{F} \times \mathbf{E} \times \mathbf{F}$ Information x is made by entity d in context k.
$perform(e, a)$	$\subseteq \mathbf{E} \times \mathbf{Y}$ entity e successfully performs specified activity a. This fluent is an sample fluent to describe trustee's performance. In applications, this fluent can replaced with any fluent representing trustee's performance.

Table 5.2: Predicates

Predicate	**Definition/Sematics**
$holds(p, s)$	$\subseteq \mathbf{F} \times \mathbf{S}$ as defined in situation calculus, fluent p holds in situation s.
$Poss(a, s)$	$\subseteq \mathbf{A} \times \mathbf{S}$ It is possible to perform action a in situation s.
$entail(q, k)$	$\subseteq \mathbf{F} \times \mathbf{F}$ Context q entails context k.

in situation calculus.

In addition, predicate *entail* is formally defined as follows.

Definition TR-2:

$$entail(q, k) \equiv \forall s, (holds(q, s) \supset holds(k, s)) \tag{5.4}$$

By this definition, we further have the following two propositions, which need to be used later. The proofs of these propositions can be found in Appendix C.

Proposition TR-1: Given any fluents p and q,

$$entail(p \dot{\wedge} q, p). \tag{5.5}$$

Proposition TR-2: Given any fluents p, q, and k,

$$entail(k, p) \wedge entail(p, q) \supset entail(k, q). \tag{5.6}$$

5.4.4 Formal Competency Questions

Given the terminology defined above, the informal competency questions can be formally represented as follows:

Q1: In a specific context, can an entity (trustor) trusts another entity (trustee) regarding what the trustee performs?

$$\forall x, trust\_p(d, e, x, k)?$$

Q2: In a specific context, can an entity (trustor) trusts another entity (trustee) regarding what the trustee believes?

$$\forall x, trust\_b(d, e, x, k)?$$

Q3: When an entity has trust in another entity's belief in a given context, and the second entity has trust in a third entity's performance (or belief) in another context, can the first entity have trust in the third entity's performance (or belief)? if so, in what context?

$$\begin{aligned}(\forall x)(holds(trust\_b(d, c, x, k), s)) \\ \wedge\ (\forall x)(holds(trust\_b(c, e, x, q), s)) \\ \supset\ (\forall x)(holds(trust\_b(d, e, x, ?r), s))?\end{aligned} \quad (5.7)$$

$$\begin{aligned}(\forall x)(holds(trust\_b(d, c, x, k), s)) \\ \wedge\ (\forall x)(holds(trust\_p(c, e, x, q), s)) \\ \supset\ (\forall x)(holds(trust\_p(d, e, x, ?r), s))?\end{aligned} \quad (5.8)$$

Q4: When an entity has trust in another entity's performance (or belief) in a specific context, given the information created by the second entity within the context, can the first entity believe this information?

$$\begin{aligned}\forall x)(holds(trust\_p(d, e, x, k), s)) \\ \wedge\ holds(made(y, e, q), s) \wedge entail(q, k\theta) \\ \supset\ holds(believe(d, k\theta \dot{\supset} y), s)?\end{aligned} \quad (5.9)$$

$$\begin{aligned}(\forall x)(holds(trust\_b(d, e, x, k), s)) \\ \wedge\ holds(believe(e, q \dot{\supset} y), s) \wedge entail(q, k\theta) \\ \supset\ holds(believe(d, k\theta \dot{\supset} y), s)?\end{aligned} \quad (5.10)$$

5.5 Axioms

5.5.1 Formal Semantics of Belief

Trust has the semantics that the trustor believes in her/his expectation on the trustee. Therefore, the representation of trust needs to be based on belief.

The logic of *knowledge* and *belief* has been studied in epistemic logic (refer to [124], [45], [187]). The systems ***T***, ***S4*** and ***S5*** are applied to knowledge, and systems ***K45*** and ***KD45*** are applied to belief. The major difference between knowledge and belief is commonly recognized as axiom ***T*** is applied to knowledge, but not to belief. In other words, knowledge must be true, but belief needs not. In terms of possible world semantics, knowledge (in a specific possible world) is defined as propositions true in all accessible possible worlds to this world. However, what is the semantics of belief in the possible world semantics?

There are many notations of knowledge and belief [124]. One approach is to define belief as the same as to define knowledge with the possible world semantics, but the accessibility relation for belief is not reflexive. Corresponding to this, the axiom ***T*** is not true for belief, but other axioms ***K***, ***D***, ***4*** and ***5*** may be applied to belief as does to knowledge. This approach raises another problem, what is the semantics of the accessibility relation for belief?

van der Hoek and Meyer [124] (pp.73) define belief as possibility, so that the semantics of the accessibility relation for belief is the same as the one for knowledge. We adopt a similar approach. Consider that belief is not necessarily true but possibly true; and something possibly true is not necessarily believed. In this perspective, we tend to define the necessary condition of belief as *possibly true*; and the sufficient conditions of belief is trust. More strictly, the sufficient conditions of belief should be trust and being consistent with this agent's knowledge. The difference between our definition of belief and van der Hoek&Meyer's model is: in van der Hoek&Meyer's model [124] (pp.73), *possibly true* is

both the necessary and sufficient conditions of belief; in our model, *possibly true* is only the necessary condition of belief, and trust is the sufficient condition of belief. Trust will be discussed later in this chapter. In the following, the logic system of belief used in this thesis is defined.

In this thesis, we use First Order Logic rather than Modal Logic to represent the logic of belief. First of all, instead of representing belief as a modal operator, we represent belief as a relational fluent, *believe(d,x)*, which means agent d believes x, and x is again a relational fluent.

The necessary condition of belief is *possibly true*, which is formally defined as the following axiom.

Axiom TR-1 (necessary condition of belief):

$$holds(believe(d, x), s) \supset (\exists s')((s, s') \in R_d \wedge holds(x, s')) \tag{5.11}$$

where R_d is the accessibility relation from the view of d,

This axiom states that if agent d believes fluent x in situation s, then x must be true in some accessible (from the view of d) possible worlds. This is because when a thing is believed, it is consistent with both what is known in current epistemic state (current world) and what is assumed in a specific accessible possible world (from the current world).

Secondly, we define the logic of belief in this thesis based on system K, which is the most general system for representing knowledge and belief in epistemic logic. In our formalization, axiom K and necessitation rule are represented as follows.

Axiom K:

$$holds(believe(d, p\dot{\supset}q), s) \supset (holds(believe(d, p), s) \supset holds(believe(d, q), s)) \tag{5.12}$$

Necessitation Rule:

$$(\forall s, holds(p, s)) \supset (\forall s', holds(believe(d, p), s')) \tag{5.13}$$

In addition, we include the following axiom in our system.

Axiom TR-2:

$$holds(believe(d, p\dot{\wedge}q), s) \equiv (holds(believe(d, p), s) \wedge holds(believe(d, q), s)) \quad (5.14)$$

In our context of trust formalization, belief is regarding believing an expected thing in the context of trust, therefore belief often needs to be represented as a conditional belief as follows,

$$holds(believe(d, c\dot{\supset}x), s).$$

where fluent c is the condition for that belief (the context of trust), and fluent x is the information created by the trusted entity, or x is a fluent to represent that someone successfully performs an activity, e.g. *perform(e,a)*, representing e successfully performs activity a.

The following proposition states that at any situation s, if an agent (d) believes a thing (x) in a condition (c) and this condition holds in that situation, then this agent believes that thing.

Proposition TR-3

$$holds(believe(d, c\dot{\supset}x), s) \wedge holds(believe(d, c), s) \supset holds(believe(d, x), s) \quad (5.15)$$

This proposition is directly obtained by *axiom K* and *modus ponens*.

The following proposition states that at any situation s, if an agent (d) believes a thing (x) in a condition (k) and there is another condition (q) that entails the first condition (k), then this agent also believes that thing in the second condition (q).

Proposition TR-4

$$holds(believe(d, k\dot{\supset}x), s) \wedge entail(q, k) \supset holds(believe(d, q\dot{\supset}x), s) \quad (5.16)$$

The proof of this proposition is presented in appendix 5.

5.5.2 Formal Semantics of Trust

As revealed in the motivating scenarios, in accordance with the types of expectancies, we classify trust into two types: *trust in performance*, and *trust in belief*. In our definition, there is no intersection between these two types. We formally define them one by one as follows.

Trust in performance

Trust in performance is the trust in what trustee performs such as the activities performed or the information created. That is to say, the expectancy (the expected thing) in trust is the activities performed or the information created

More accurately, *trust in performance* means that the trustor believes that (1) a specific action is successfully performed by the trustee as committed to, which meets a certain quality standards; or (2) the information created by the trustee is true.

In this thesis, we will use a relational fluent to represent the expectancy in a trust, no matter this expectancy is a successfully performed activity or the information created being true. In this way, *Trust in performance* can be formally defined as follows.

Axiom TR-3 (formal semantics of *trust in performance*):

$$holds(trust\_p(d, e, x, k), s) \equiv$$
$$\forall q, (holds(made(x, e, q), s) \wedge entail(q, k)$$
$$\supset holds(believe(d, k \dot{\supset} x), s)) \quad (5.17)$$

In this axiom (5.17), the expected thing (called expectancy) is fluent x; $believe(d, k \dot{\supset} x)$ represents that d believes x to be true when context k is true. In other words, d believes x in context k.

To express such a trust relationship in practice, variable d will be bound to a trustor; variable e will be bound to a trustee; variable x will be bound to a fluent representing

the information created by trustee; variable k will be bound to a fluent representing the context of trust.

There are two different contexts in trust. q is the context for performing / making x, called the context of performance; and k is the context for using x, called the context of trust. When the expectancy (x) is a piece of information, the context to create this information and the context to use it usually are different; when the expectancy is performing an activity, these two contexts may overlap in the circumstance in which the activity is performed. However, the trustor's concerns such as goals and utilities may be quite different from trustee's concerns about the completion of this activity. For this reason, these two contexts are also different in the latter case.

As addressed earlier, the expectancy of trust can be the activities performed or the information created. When the expectancy is a piece of information created, the information is directly represented as a fluent to replace variable x in the axiom; when the expectancy is an activity successfully performed by the trustee[1], this expectancy is also represented as a fluent. We use fluent "*perform(e,a)*", which represents that trustee e successfully performs activity a, to represent that expectancy. In applications, it can be replace as needed. After replace x in Axiom TR-3 with "*perform(e,a)*", the axiom is in the following form:

Axiom TR-3(b) (trust in performing activities):

$$holds(trust\_p(d, e, perform(e, a), k), s) \equiv$$
$$\forall q, (holds(made(perform(e, a), e, q), s) \wedge entail(q, k)$$
$$\supset holds(believe(d, k \dot{\supset} perform(e, a)), s)) \quad (5.18)$$

This form of axiom can be read as that at any situation s, trustor d trusts trustee e on successfully performing activity a in context k is logically equivalent to that if e

[1]The activity performed by the trustee is not an "action" modeled in situation calculus, so that this activity is represented as a term.

commits to doing a in context q, and context q is within k, then d believes in that e successfully performs a in context k. Here, the straightforward semantics of fluent "$made(perform(e,a),e,q)$" is that fluent (information) "$perform(e,a)$" is made by e in context q. So, the semantics of this fluent should be understood as in context q, e commits to successfully perform activity a .

The following are two examples for this representation.

Example 1 trust in information created. Ben, a customer of Amazon, trusts Amazon regarding the message that his order status is "Shipped" in the context that Ben made his order at Amazon. This trust relationship can be represented as follows:

$$holds(trust\_p(Ben, Amazon, order\_status(\#102579, Shipped), \\ order\_at(\#102579, Amazon) \dot{\supset} order\_of(\#102579, Ben), S_0)$$

According to the axiom TR-3, this means that for any context q, if Amazon creates information

$$order\_status(\#102579, Shipped)$$

in q, and q entails context

$$order\_at(\#102579, Amazon) \dot{\wedge} order\_of(\#102579, Ben),$$

then Ben believes that information. The formal representation of this meaning is as follows:

$$(\forall q)(holds(made(order\_status(\#102579, Shipped), Amazon, q), S_0) \\ \wedge entail(q, order\_at(\#102579, Amazon) \dot{\wedge} order\_of(\#102579, Ben)) \\ \supset holds(believe(Ben, order\_at(\#102579, Amazon) \dot{\wedge} order\_of(\#102579, Ben) \\ \dot{\supset} order\_status(\#102579, Shipped)), S_0))$$

q can be any context that entail the context of the trust, for example,

$$order\_at(\#102579, Amazon)\dot{\wedge}order\_of(\#102579, Ben)$$
$$\dot{\wedge}order_content(\#102579, book(\text{“}KnowledgeinAction\text{”}, 1))$$
$$\dot{\wedge}shipped\_by(\#102579, UPS)\dot{\wedge}order\_status(\#102579, Shipped).$$

Example 2. Ben trusts Amazon regarding getting refund for an order in the context that he wants get refund for some reasons and the order meets Amazon's returns policy. This trust relationship can be represented as follows:

$$holds(trust\_p(Ben, Amazon,$$
$$perform(Amazon, refund(\#102579)),$$
$$refund\_asked(\#102579)\dot{\wedge}meets\_r\_p(\#102579)),$$
$$s)$$

Different from example 1, the expectancy in this trust relationship is that Amazon refunds Ben's order.

$$perform(Amazon, refund(\#102579)).$$

Trust in belief

Trust in belief is the trust placed on what trustee believes.

The semantics of *trust in belief* is that the trustor believes a thing believed by the trustee in a context within the trustor's context of trust. This semantics can be formally defined in the following axiom.

Axiom TR-4 (formal semantics of *trust in belief*):

$$holds(trust\_b(d, e, x, k), s) \equiv$$
$$\forall q, holds(believe(e, q\dot{\supset}x), s) \wedge entail(q, k)$$
$$\supset holds(believe(d, k\dot{\supset}x), s) \quad (5.19)$$

Example 4. In the context of our motivating example, F trusts what P believes regarding the quality of a porcelain product, e.g. "*TeaSet-J1106b*". This trust can be represented as follows.

$$\begin{aligned} &holds(trust\_b(F, P, qual\_grade(TeaSet - J1106b, A), \\ &\qquad in\_topic(qual\_grade(TeaSet - J1106b, A), \\ &\qquad Porc\_Qual)), s), \end{aligned} \tag{5.20}$$

where $qual\_grade((TeaSet - J1106b, A)$ represents the quality grade of product *TeaSet-J1106b* is A.

5.5.3 Reasoning with Trust Relationships

The defined formal semantics of trust can be used to derive the belief in the information created or an activity performed by a trusted entity. The following two theorems provide rules for such type of inference.

Theorem TR-1 states that if a trustor (d) trusts a trustee (e) in every thing (x) made by the trustee in a given context (k), and there is a thing (y) made by the trustee in a context (q) covered by the given context of trust (k), then the trustor believes this thing (y) in the given context.

Theorem TR-1.

$$\begin{aligned} &(\forall x)(holds(trust\_p(d, e, x, k), s)) \\ &\qquad \wedge holds(made(y, e, q), s) \wedge entail(q, k) \\ &\qquad \supset holds(believe(d, k \dot{\supset} y), s) \end{aligned} \tag{5.21}$$

The proof of this theorem is given in Appendix. This theorem needs to be used to derive belief in performance from trust in performance.

Theorem TR-2 states that if a trustor (d) trusts a trustee (e) in every thing (x) believed by the trustee in a given context (k), and there is a thing (y) that the trustee

believes in a context (q) covered by the given context of trust (k), then the trustor also believes this thing (y) in the given context.

Theorem TR-2.

$$(\forall x)(holds(trust\_b(d,e,x,k),s)) \\ \wedge holds(believe(e,q\dot{\supset}y),s) \wedge entail(q,k) \\ \supset holds(believe(d,k\dot{\supset}y),s) \quad (5.22)$$

The proof of this theorem is given in Appendix 5

Theorem TR-3 reveals that if d trust e on thing x in context k, then d also trusts e on x in a stricter context (that satisfies k).

Theorem TR-3.

$$holds(trust\_p(d,e,x,k),s) \wedge entail(q,k) \supset holds(trust\_p(d,e,x,q),s) \quad (5.23)$$

$$holds(trust\_b(d,e,x,k),s) \wedge entail(q,k) \supset holds(trust\_b(d,e,x,q),s) \quad (5.24)$$

The proof of this theorem is given in Appendix.

5.6 Transitivity of Trust

Trust propagation in social networks is based on the assumption that trust is transitive. However, trust is not always transitive. For example, A trusts B to access A's resources, and B trusts C to access B's resources, but these do not necessarily imply that A trusts C to access A's resources. The interesting question here is what type of trust is transitive.

The following theorem answers this question.

Theorem TR-4 (Transitivity of trust in belief).

(a) In any situation s, if entity d trusts entity c on everything which c believes in context k, and c trusts entity e on everything which e believes in context q, then d trusts

e on everything which e believes in the conjunction of the contexts k and q.

$$(\forall x)(holds(trust\_b(d, c, x, k), s)) \wedge (\forall x)(holds(trust\_b(c, e, x, q), s)) \supset (\forall x)(holds(trust\_b(d, e, x, k\dot{\wedge}q), s)) \quad (5.25)$$

(b) In any situation s, if agent d trusts agent c on everything which c believes in context k, and c trusts agent e on everything which e performs in context q, then d trusts e on everything which e performs in the conjunction of contexts k and q.

$$(\forall x)(holds(trust\_b(d, c, x, k), s)) \wedge (\forall x)(holds(trust\_p(c, e, x, q), s)) \supset (\forall x)(holds(trust\_p(d, e, x, k\dot{\wedge}q), s)) \quad (5.26)$$

This theorem shows that ***trust in belief*** **is transitive;** ***trust in performance*** **can be derived by using** ***trust in belief***. However, ***trust in performance*** **by itself is not transitive**, which can be justified by examples such as the access control example in the beginning of this section, as well as case 3 of the motivating scenarios.

Trust in performance also can be derived by *system trust*. *System trust* makes *trust in performance* able to be transferred from an organization or a group to its members. Case 2 in the motivating scenarios is an example of using system trust. For length limitation, system trust is not covered in this thesis. A simple solution of system trust can be found in [89].

Theorem TR-4 together with system trust also described three conditions and forms of trust propagation: (1) Entity d has *trust in belief* relationship with entity c, c has *trust in belief* relationship with entity e, then we can derive that d has *trust in belief* relationship with e; (2) d has *trust in belief* relationship with c, c has *trust in performance* relationship with e, then it can be derived that d has *trust in performance* relationship with e; (3) d has *trust in performance* relationship with o, o is a system (organization

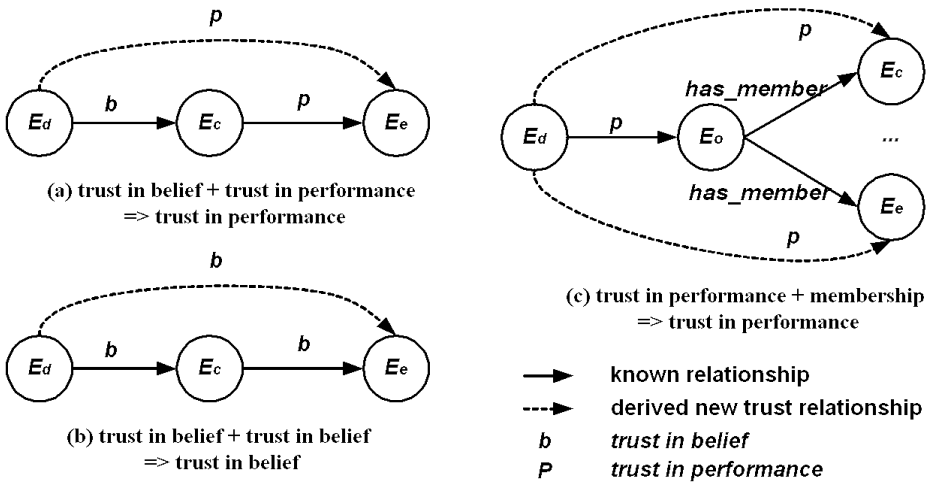

Figure 5.1: Three forms and conditions of trust propagation in a social networks

or group) and has member e, then it can be derived that d has *trust in performance* relationship with e. These are illustrated in figure 5.1. For simplicity, we assume the context of trust is the same and omitted in the figure.

5.7 Sources of Trust

In the earlier sections, we discussed the semantics of trust or what trust means and the transitivity of trust. This subsection discusses where trust comes from.

Trust comes from three types of sources: (1) the trust built up from the experience of interaction between trustor and trustee, called *inter-individual trust* or *direct trust*[2]; (2) trust derived through trust propagation in social networks, called *relational trust* or "social networks based trust"; (3) trust via the trust placed on the stable or predictable functions / behaviors of a system (e.g. an organization, a culture, or a group of individuals with certain common features), called *system trust*, typically including institutional based trust, professional membership based trust, as well as characteristics based trust. The

[2] *Inter-individual trust* and *direct trust* have the same meaning and they are exchangeable in this thesis.

above (2) and (3) are also called indirect trust. Direct trust is essential; indirect trust needs to be derived from direct trust.

Previously, fluents *trust_b* and *trust_p* are used to represent trust relationships that may be inter-individual (or direct) trust relationships and may be derived (or indirect) trust relationships. At that time, we need not to discern the difference.

In order to discuss trust propagation in social networks, in which direct trust relationships are represented with arcs, and indirect trust relationships are represented by the paths comprising certain those arcs, we need to tell the difference of direct and indirect trust right now.

Inter-individual trust relationships are represented with fluents:

- *has_p_tr(d,e,x,k)*, which represents trustor d has *trust in performance* type of inter-individual trust relationship with trustee e in context k;

- *has_b_tr(d,e,x,k)*, which represents trustor d has *trust in belief* type of inter-individual trust relationship with trustee e in context k.

Inter-individual trust has the same semantics as the general concept of trust. Therefore, the general semantics of trust is the necessary condition of inter-individual trust as described in the following axiom. The sufficient condition for inter-individual trust depends on how trust is built up in the process of interaction among entities. Many researchers have studied this problem (refer to chapter 2). In this thesis, we will not discuss how inter-individual trust relationships are built up; instead we only assume that entities in social networks have developed their inter-individual trust relationships, from which the indirect trust relationships are derived.

Axiom TR-5.

$$holds(has\_p\_tr(d, e, x, k), s) \supset holds(trust\_p(d, e, x, k), s) \tag{5.27}$$

$$holds(has\_b\_tr(d, e, x, k), s) \supset holds(trust\_b(d, e, x, k), s) \tag{5.28}$$

The sufficient condition for inter-individual trust depends on how trust is built up in the process of interaction among entities. In this thesis, we will not discuss how inter-individual trust relationships are built up; instead we only assume that entities in social networks have developed their inter-individual trust relationships, from which the indirect trust relationships are derived.

5.8 Trust Propagation in Social Networks

This section discusses the graph representation of trust propagation in social networks and its connection with the proposed trust ontology.

By graph representation, the problem to infer an indirect trust relationship by using the proposed trust ontology can be equivalently transformed into a more straightforward approach – the problem of searching for a trust path in a trust network.

5.8.1 Trust Networks

A *trust network* is a subgraph of a social network. A ***social network*** can be regarded as a directed graph with labeled arcs where the vertices are entities such individuals and organizations in society, and the arcs are various social relationships. In the context of trust, we only concern a special type of subgraphs of social networks, called ***trust networks***, in which arcs represent inter-individual (or direct) trust relationships. An arc from vertex d to vertex e represents that d directly trusts e.

A trust network can be defined as a directed graph as follows.

$$\mathcal{TN} = \mathcal{G}(\mathcal{V}, \mathcal{A}) \tag{5.29}$$

$$\mathcal{A} = \mathcal{T}^b \cup \mathcal{T}^p \tag{5.30}$$

$$\mathcal{T}^b \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{K} \tag{5.31}$$

$$\mathcal{T}^p \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{K} \tag{5.32}$$

where, $\mathcal{V}$ is the set of vertices (all entities in social networks, i.e. individuals or organizations in consideration; $\mathcal{V}$ is equivalent to $\mathbf{E}$ as defined in Chapter 6); $\mathcal{A}$ is the set of arcs; to define $\mathcal{A}$, $\mathcal{T}^b$ is defined as a relation, called *trust in belief* relation; $\mathcal{T}^p$ is also a relation, called *trust in performance* relation; $\mathcal{K}$ is the set of contexts; thus, each arc in $\mathcal{A}$, e.g. (d, e, t, k), is from one vertex (d) to another (e) with a pair of labels *(t,k)* in which t is "b" representing *trust in belief* or "p" representing *trust in performance*, k is the context of trust. In this way, each arc represents an inter-individual trust relationship: entity d trusts entity e regarding e's belief/performance in context k. In the following discussion, T^b and T^p are also used as predicates to represent a trust relationship, e.g. $T^b(d, e, k)$, representing d trusts in e regarding belief in context k.

The terminology used in this section is summarized in table 5-3.

In this representation, the representation of context remains open. It could be defined in the way that accommodates the intended application. For example, a context could be defined as a knowledge field as we did in KP.

Definition TR-3 (*trust path*): In a trust network, a path is called a *trust path*, if: (1) all the arcs except the last one are *b-arcs*; (2) there are no circles in the path; (3) the "conjunction" of all contexts in the path is consistent. Here, *b-arcs* are arcs with label "b"; *p-arcs* are arcs with label "p".

A trust path (with length greater than one) is a visual and straightforward representation of a derived trust relationship.

In social networks, if an entity knows another entity enough, the first entity has inter-individual trust relationships with the second, that is, there are arcs from the first to the second entity; otherwise, there is no inter-individual trust relationships between these two entities; however, if there is a trust path between them, then a indirect trust relationship can be derived.

Trust Inference Rule: *In a trust network, if an entity (represented by a vertex) has a trust path (refer to def. 1) to another entity (represented by another vertex), then*

Table 5.3: Notation for trust networks

Notation	**Definition**
$\mathcal{TN}$	the defined trust network.
$\mathcal{V}$	the set of vertices in $\mathcal{TN}$
$\mathcal{A}$	the set of arcs in $\mathcal{TN}$. $\mathcal{A} = \mathcal{T}^b \cup \mathcal{T}^p$.
$\mathcal{K}$	the set of contexts of trust.
$\mathcal{T}^b$	$\subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{K}$ a relation of *trust in belief*.
$\mathcal{T}^p$	$\subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{K}$ a relation of *trust in performance*.
$T^b(d, e, k)$	$\subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{K}$ a predicate, representing that d trust in e regarding belief in context k.
$T^p(d, e, k)$	$\subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{K}$ a predicate, representing that d trust in e regarding performance in context k.
S	a specific situation in situation calculus, at which all held trust relationships forms the trust network.
$\mathcal{TN}(\mathcal{S})$	the defined trust network at situation S.
$\mathcal{A}(\mathcal{S})$	the set of arcs in the deterministic trust network at situation S.
$\mathcal{T}^b(S)$	$\subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{K}$ a relation of *trust in belief* at situation S.
$\mathcal{T}^p(S)$	$\subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{K}$ a relation of *trust in performance* at situation S.

this entity has a trust relationship with that entity in the context of the "conjunction" of all contexts in the path. The trust type of this trust path depends on the trust type of the last arc. In formal,

$$T^b(e_1, e_2, k_1) \wedge T^b(e_2, e_3, k_2)... \wedge T^p(e_{n-1}, e_n, k_{n-1}) \supset T^p(e_1, e_n, k_1 \wedge k_2... \wedge k_{n-1}) \quad (5.33)$$

$$T^b(e_1, e_2, k_1) \wedge T^b(e_2, e_3, k_2)... \wedge T^b(e_{n-1}, e_n, k_{n-1}) \supset T^b(e_1, e_n, k_1 \wedge k_2... \wedge k_{n-1}) \quad (5.34)$$

The above graph representation of trust networks can be used independently to the proposed trust ontology. The only two requirements coming from the logical theory are: (1) to discern *trust in belief* and *trust in performance*; (2)each trust relationship is associated with a context. These requirements are also one of the major difference between the trust networks defined above and most trust network models that have appeared in the literature.

5.8.2 Formal Semantics of Trust Networks

Although the above graph representation can be used independently to the proposed trust ontology, the latter provides formal semantics and theoretical foundation for the former. In the remainder of this section, we discuss the connection between the presented trust network model and the proposed trust ontology.

In real world, trust networks are dynamic rather than static. In the real world, trust networks change over time, because there are always entities that are adjusting their trust relationships with the increasing experience of interactions with others; from the view of the logical theory of trust in situation calculus, trust networks dynamically change with the changing of situations. To address this feature, we define a trust network as a snapshot of all inter-individual trust relationships in a specific situation.

By the logical theory, the relations $\mathcal{T}^b$ and $\mathcal{T}^p$, as well as trust network $\mathcal{TN}$ at

situation S can be defined as follows.

$$\mathcal{T}^b(S) = \{(d, e, k)|\forall x, holds(has\_b\_tr(d, e, x, k), S) \tag{5.35}$$

$$\mathcal{T}^p(S) = \{(d, e, k)|\forall x, holds(has\_p\_tr(d, e, x, k), S) \tag{5.36}$$

$$\mathcal{TN}(S) = \mathcal{G}(\mathcal{V}, \mathcal{A}(S)) \tag{5.37}$$

$$\mathcal{A}(S) = \mathcal{T}^b(S) \cup \mathcal{T}^p(S) \tag{5.38}$$

Predicates T^p and T^b have the following semantics:

$$T^p(d, e, k) = \forall x, holds(trust\_p(d, e, x, k), S); \tag{5.39}$$

$$T^b(d, e, k) = \forall x, holds(trust\_b(d, e, x, k), S); \tag{5.40}$$

In this way, the transitivity of *trust in belief* relation, the soundness and completeness of trust inference rule in trust networks can be guaranteed by the trust ontology.

Axiom TR-6 (Causal Completeness Assumption): axiom TR-5, theorems TR-3 and TR-4 are all conditions under which a trust relationship can be derived.

Theorem TR-5: For every trust path in a trust network, the corresponding trust relationship is valid by the trust ontology.

The proof of theorem can be found in Appendix C.

Theorem TR-6: For every valid trust relationship by the trust ontology, there exists at least a trust path within the context of the trust and with finite length in a trust network.

The proof of theorem can be found in Appendix C.

The above two theorems show that the graph model of trust networks and the proposed trust ontology are equivalent regarding trust reasoning using social networks. From theorem TR-5, the trust relationship derived from a trust path is valid by the trust ontology. This result shows the **soundness** of trust propagates along a trust path in social networks. From theorem TR-6, every valid trust relationship that can be derived from the trust ontology corresponds to at least one trust path in a trust network. This result shows the **completeness** of trust networks.

Interestingly, based on these theorems, the problem of inferring a trust relationship by using trust ontology can be equivalently transformed into the problem of searching a trust path in a trust network. The developing techniques for search social networks can be applied to find trust paths in trust networks.

The model proposed in this chapter is a deterministic model. Although this model can be used in real trust judgments, it is important to extend the model to uncertain model, because uncertainty widely exists in trust problems, especially in the cyberspace. We extend our model to uncertain model in papers [94] [88].

5.9 Application Examples

In the following, to demonstrate the potential uses of the proposed trust ontology in trust judgments using social networks on the web, we provide two examples: (1) "web of trust" in PGP; (2) trust judgments in web services based e-business.

5.9.1 Example 1: Web of Trust in PGP

PGP [195] is a public key cryptosystem used for encryption and digital signatures. *Web of trust* is a trust model used in PGP to validate public key. Let us consider the following story.

John received a message with the public key certificate of the sender Bill. In order to verify whether the message is authentic, John needs to validate this public key certificate first. However, John does not know Bill. John needs to validate Bill's public key certificate through his trusted friends.

First, let us look into Bill's public key certificate. This certificate consists of (1) Bill's public Key; (2) Bill's ID information; (3) the digital signature signed by Bill himself, and the digital signatures signed by a set of introducers who know Bill and have validated Bill's public key.

One of these introducers is David. That is to say, David has validated Bill's public key certificate, and he believes in this certificate.

John's public keyring consists of a set of keys in which John trusts the owners of these keys to validate other people's public key certificates. One of the key owners is Alice, in the term of our logic model, John trusts Alice on what Alice believes in the context of public key certificate.

Assume Alice does not know Bill either, but Alice has David's public key in her public keyring, that is to say, Alice trusts David in what David believes in the context of public key certification.

Now, John trusts Alice regarding what Alice believes in public key certification, Alice trusts David regarding what David believes in the same issue, and David believes the validity of Bill's public key and signed Bill's public key certificate as an introducer. In this way, John could validate the authenticity of Bill's public key.

In the following, we represent the facts stated above with our proposed logic model of trust. (We only consider "complete trust". The trust with uncertainty will be addressed in the next chapter.) The following fluents need to be defined in this application.

$pk(u)$: the public key of PGP user u;

$id\_info(u)$: the id information of user u;

$owner\_of(id\_info(u), pk(u))$: the owner of public key $pk(u)$ is the individual described by $id\_info(u)$. This is the major information described by public key certificate;

$is\_a(x, PK\_Cert)$: x is a public key certificate.

Assume that at situation s_1, John decides to validate information $owner\_of(id\_info(Bill), pk(Bill))$ created by Bill.

We represent the facts stated earlier first. All free variable are universally quantified in largest scope.

(Fact-1) John has Alice's key in his public keyring.

$$(\forall x) holds(trust\_b(John, Alice, x, is\_a(x, PK\_Cert)), s_1). \tag{5.41}$$

(Fact-2) Alice has David's key in her public keyring.

$$(\forall x)holds(trust\_b(Alice, David, x, is\_a(x, PK\_Cert)), s_1). \tag{5.42}$$

(Fact-3) David is a introducer of Bill's public key certificate. That is to say, David has validated Bill's public key certificate and he believes the information $owner\_of(id\_info(Bill), pk(Bill)$ is true.

$$holds(believe(David, is\_a(owner\_of(id\_info(Bill), pk(Bill)), PK\_cert) \dot{\supset} owner\_of(id\_info(Bill), pk(Bill))), s_1). \tag{5.43}$$

(Fact-4) $owner\_of(id\_info(Bill), pk(Bill))$ is a PK_Cert.

$$holds(is\_a(owner\_of(id\_info(Bill), pk(Bill)), PK\_Cert), s_1) \tag{5.44}$$

By necessitation rule, everyone believes the above true proposition, so John does.

$$holds(believe(John, is\_a(owner\_of(id\_info(Bill), pk(Bill)), PK\_Cert)), s_1) \tag{5.45}$$

Now we illustrate the logical process of validating this public key. By the definition of entail, we have $entail(p, p)$, in particular here,

$$holds(entail(is\_a(owner\_of(id\_info(Bill), pk(Bill)), PK\_Cert)), is\_a(owner\_of(id\_info(Bill), pk(Bill)), PK\_Cert))) \tag{5.46}$$

From this proposition, facts (3),(2), and Theorem TR-2, we have

$$holds(believe(Alice, is_a(owner\_of(id\_info(Bill), pk(Bill)), PK\_cert) \dot{\supset} owner\_of(id\_info(Bill), pk(Bill))), s_1). \tag{5.47}$$

Applying Theorem TR-2 again to above two propositions and fact (1), we further have

$$holds(believe(Alice, is_a(owner\_of(id\_info(Bill), pk(Bill)), PK\_cert) \dot{\supset} owner\_of(id\_info(Bill), pk(Bill))), s_1). \tag{5.48}$$

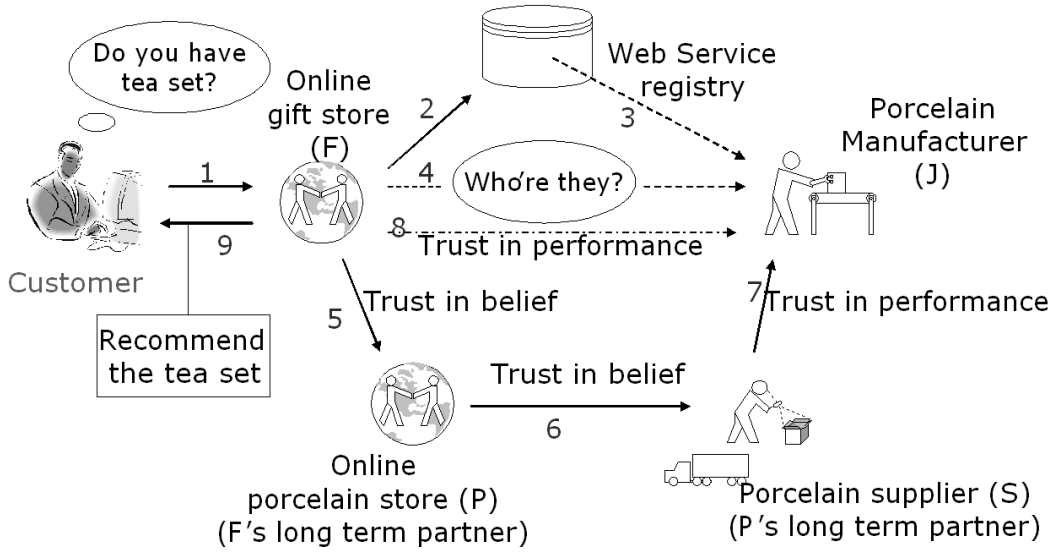

Figure 5.2: example: trust in a social network of web services and users

From this proposition and ((5.45)), applying proposition TR-3, we have

$$holds(believe(Alice, owner\_of(id\_info(Bill), pk(Bill))), s_1). \tag{5.49}$$

This proposition shows that after validation process John reaches a conclusion that Bill's public key certificate is authentic.

5.9.2 Example 2: Trust in Web Services

We apply the trust ontology to the trust j udgment problem in web services base e-business introduced in the motivating scenarios.

The trust relationships in case 1 of the motivating scenarios is formally represented as follows.

(1) F trusts in P's belief on porcelain product quality.

$$holds(trust\_b(F, P, x, in\_topic(x, Porc\_Qual)), s) \tag{5.50}$$

Here x is a free variable, so it universally quantified in the largest scope.

(2) P trusts in S's belief on porcelain product quality.

$$holds(trust\_b(P, S, x, in\_topic(x, Porc\_Qual)), s) \quad (5.51)$$

Similar to (1), x is a free variable.

(3) S trusts in J's performance on high quality porcelain product manufacture.

$$holds(trust\_p(S, J, perform(J, makeHQPorcln(x)), in\_topic(perform(J, makeHQPorcln(x)), Porcln\_Qual)), s) \quad (5.52)$$

Here, term $makeHGPorcel(x)$ represents J's performance on high quality porcelain product manufacture.

Now, we use the above facts and the proposed trust ontology to answer whether F trusts in J's performance on high quality porcelain product manufacture.

From (5.50), (5.51) and Theorem TR-4a (5.25), we have,

$$holds(trust\_b(F, S, x, in\_topic(x, Porc\_Qual)), s) \quad (5.53)$$

From (5.53), (5.52) and Theorem TR-4b (5.8), we have,

$$holds(trust\_p(F, J, makeHQPorcln(x), in\_topic(perform(J, makeHQPorcln(x)), Porcln\_Qual)), s) \quad (5.54)$$

This formula gives the answer that F can trust in J's performance that J's porcelain products have high quality. Because of this trust, F presents to its customers J's products.

The examples show that the proposed trust ontology can be used in trust judgments using social networks. This ontology can be used as a kernel logic part in specific trust models for particular domains. In practice, a specific trust judgment model in a particular domain can be built by incorporating this trust ontology and domain-dependent knowledge.

5.10 Summary and Discussion

Making trust judgments by using social networks is a promising approach for addressing the trust problem on the Web. The necessary condition for trust propagation in social networks is that trust is transitive. However, a formal study on this issue is missing.

Many formal trust models have been proposed (refer to Chapter 2), but the transitivity of trust has not been studied in formal manner. Although [1] and [99] argued that "recommendation trust" is transitive, but neither gives a formal description. In addition, "recommendation trust" is a specific case of *trust in belief*; the latter is more general.

In this chapter, we created a logical theory of trust in the form of ontology that gives a formal and explicit specification for the semantics of trust. From this formal semantics, we identified two types of trust – *trust in belief* and *trust in performance*, and we proved the transitivity of *trust in belief* and the conditions for trust propagation in social networks. These results answered the questions of "is trust transitive, what types of trust are transitive and why", and provided a theoretical evidence for trust propagation in social networks.

On the issue of knowledge representation, our model has three advantages: (1) we represent trust based on belief, a well studied concept in AI, which makes our model established on a concrete ground; (2) by using situation calculus as representation language, we are able to represent the context of trust as fluents. In this way, we found a solution to formally represent the context of trust; (3) the representation of trust in situation calculus also contributes to situation calculus for the language to describe trust among multiple agents. As we know, this is the first proposed model of trust in situation calculus.

To facilitate trust reasoning in an easier way, from proposed trust ontology, we also constructed a trust networks model, a straightforward representation for trust propagation in social networks. The soundness and completeness of the trust networks were proved.

As illustrated in examples, the proposed trust ontology can be used as a logical tool to support trust judgements using social networks on the Web.

The proposed logic model of trust is general. There may be many different web implementations, but the logic behind these implementations is the same.

Chapter 6

Distributed Trust Reasoning

Most proposed social networks based trust models assume that the entity making trust judgment is able to access to all personal trust relationships of the visited entities in social networks in a search. This assumption is not practical for the reason of privacy. In other words, people or organizations usually do not feel like publishing on the Web their private data like personal trust relationships.

To solve the problem, we propose a social network-based distributed trust reasoning model and constructs this model in situation calculus. With the model, the questioning entity requests its trusted friends to tell it whether they trust the questioned entity in a specific context; the friends may ask their friends; if any trusted friend trusts, by the transitivity of trust, the entity can trust either; each entity in social networks needn't to publish her/his/its private inter-individual trust relationships on the Web; instead each entity only answers the queries from the entities, whom this entity feels like responding, about whether this entity trusts a questioned entity in a given context. In this way, privacy can be better protected in trust reasoning using social networks. In addition, technically, the distributed trust reasoning searches a trust path in a parallel search method, so that, it is more efficient.

In this chapter, we construct a social network-based distributed trust reasoning model

in situation calculus. First, a small set of actions necessary for trust related communication in distributed trust systems are specified; secondly, the successor state axioms for trust related primary fluents are given; then, an example is presented to illustrate how the proposed distributed trust reasoning in situation calculus works; finally, a summary is given.

6.1 Terminology

In the following discussion, we assume that each agent (or, entity, as addressed earlier) works with a situation calculus system, and each agent has its own knowledge base.

We identify the following actions directly related to trust judgments:

$$request(e, query(e, e', q)),$$

$$checkAnswer(e, query(e, e', q), w),$$

$$acceptQ(e', query(e, e', q)),$$

$$replyQ(e', query(e, e', q), w).$$

They are summarized in table 7.1. Actions "*request*" and "*checkAnswer*" are conducted by the requesting agent; actions "*acceptQ*" and "*replyQ*" are conducted by the requested agent.

Except fluents defined earlier in table 6.1 and 6.2 of Chapter 6, we also need some new fluents for distributed trust reasoning. They are summarized in the table 7.2.

6.2 Actions, Preconditions and Effect Axioms

In the following, we discuss each action and its precondition axiom and effect axiom. The precondition of an action is the requirements that must be satisfied whenever the action can be executed in the current situation [150]; the effect axioms describe how actions affect the fluents, i.e. the properties of the world in modeling.

Table 6.1: Actions

Action	Definition/Sematics
$request(e, query(e, e', q))$	$\subseteq \mathbf{E} \times \mathbf{D}$ agent e, requests agent e' whether fluent q holds. $q = trust\_p(e, e', x, k)$, or $trust\_b(e, e', x, k)$;
$checkAnswer(e, query(e, e', q), w)$	$\subseteq \mathbf{E} \times \mathbf{D} \times \mathbf{D}$ agent e, gets the answer (w) for query $query(e, e', q)$.
$acceptQ(e', query(e, e', q))$	$\subseteq \mathbf{E} \times \mathbf{D}$ agent e', accepts the query from agent e.
$replyQ(e', query(e, e', q), w)$	$\subseteq \mathbf{E} \times \mathbf{D} \times \{Yes, No\}$ agent e', replies the query from agent e with w.

6.2.1 Action: $request(e, query(e, e', q))$

As shown in figure 6.1, action *request(e,query(e,e',q))* represents that agent e makes a query *query(e,e',q)* to agent *e'*, to request whether fluent q can be proved by *e'*. Generally, the question to be asked may be any fluent the requester wants to know from the requested agent, but we limit questions to only two types of fluents:

$$trust\_b(e', e_3, x, k),\ trust\_p(e', e_3, x, k),$$

representing questions whether the recipient (e') trusts another given agent (e_3) regarding what the given agent (e_3) believes or performs in a given context (k).

$query(e, e', q)$ is a functional fluent mapping to an object representing a query used in communication between agents.

The action of *request* can be caused by the condition that the questioning agent has a task to prove a fluent q, and q does not hold in this agent's KB in current situation. However, this condition is not the necessary condition for an agent to request.

Table 6.2: Fluents

$has\_query(query(e, e', q))$	$\subseteq \mathbf{D}$ Relational fluent. Query from e to e', *query(e,e',q)*, has been made. In other words, the query object *query(e,e',q)* has been created.
$has\_answer(query(e, e', q), w)$	$\subseteq \mathbf{D} \times \mathbf{D} \times \mathbf{D}$ Relational fluent. Query from e to e', *query(e,e',q)*, has answer w.
$has\_task(e, e', q)$	$\subseteq \mathbf{E} \times \mathbf{E} \times \mathbf{F}$ Relational fluent. Entity e has task to answer question q from entity q'. More exactly, e has been asked by entity q' whether fluent q can be proved.
$query(e, e', q)$	$\mathbf{E} \times \mathbf{E} \times \mathbf{F} \rightarrow \mathbf{F}$ Functional fluent, represents a query object created by requesting entity e, the requested entity is e', and the questioned fluent is q.

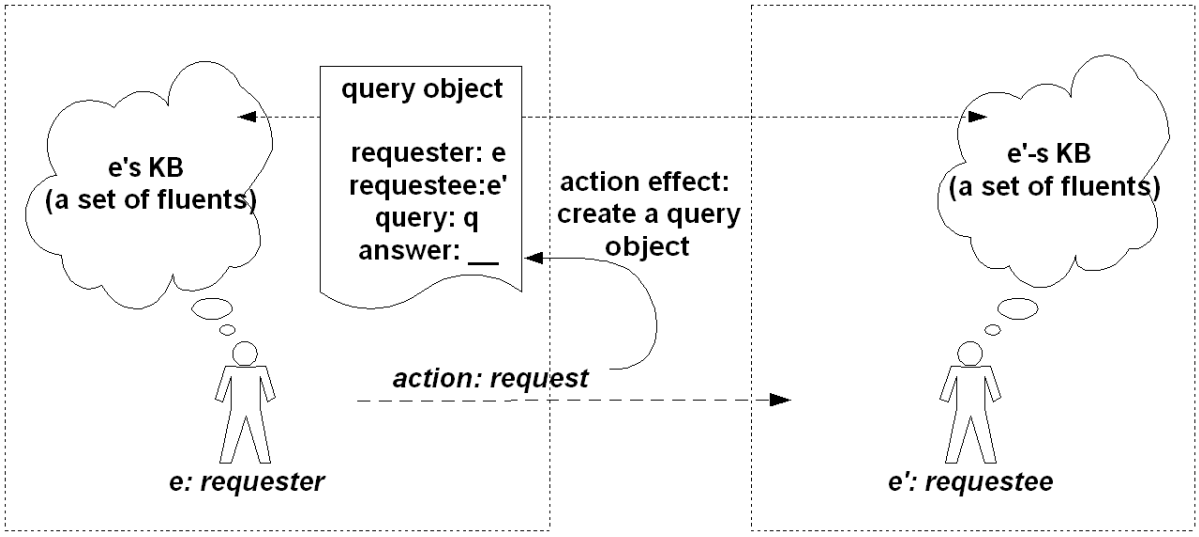

Figure 6.1: Action: request

An agent is always able to request another agent, in other words, no condition is necessary for an agent to request others, so the precondition of this action is always true.

Axiom (precondition of action *request*):

$$Poss(request(e, query(e, e', q)), s). \tag{6.1}$$

After action "request", an object $query(e, e', q)$ is created. This query object exists (or is visible) in the worlds of both sides of communication.

Axiom (effect of action *request*):

$$a = request(e, query(e, e', q)) \supset holds(has\_query(query(e, e', q)), do(a, s)). \tag{6.2}$$

Relational fluent $has\_query(query(e, e', q))$ represents that query *query(e,e',q)* has been made, or the query object *query(e,e',q)* has been created.

6.2.2 Action: $acceptQ(e', query(e, e', q))$

Action $acceptQ(e', query(e, e', q))$ represents action by which agent *e'* checks and accepts query *query(e,e',q)*. This action is illustrated in figure 6.2.

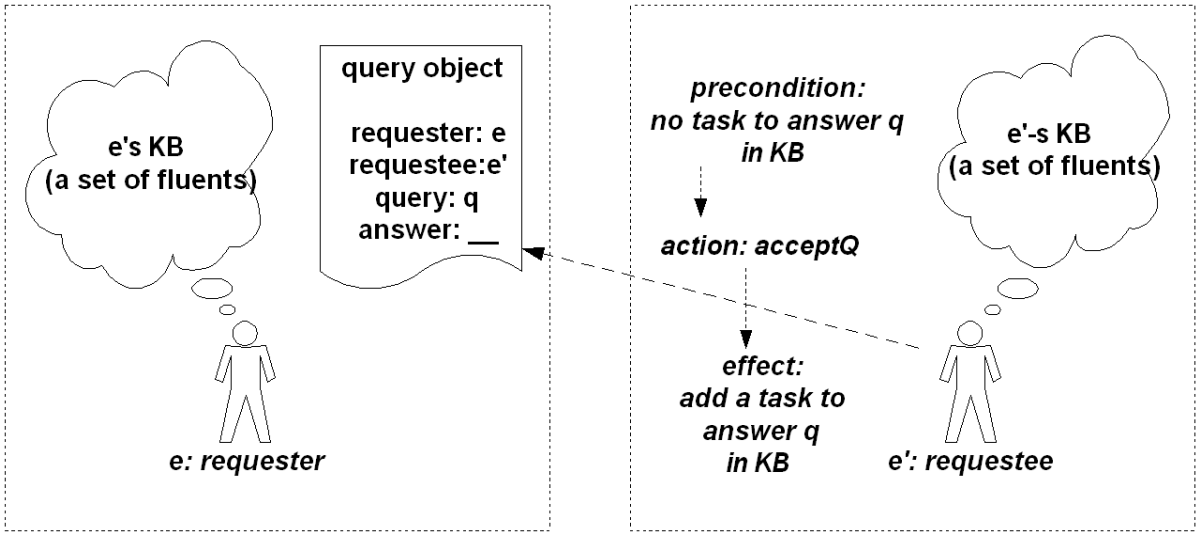

Figure 6.2: Action: acceptQ

When an agent receives a query, to accept this query, the requested agent should not be in a process to answer the same question.

Axiom (precondition of action *acceptQ*):

$$Poss(acceptQ(agt, query(e, agt, q)), s) \equiv \neg holds(has\_task(agt, , q), s). \quad (6.3)$$

This precondition prevents circles in a trust path by checking whether the agent already has a task to answer the same question.

After the recipient accepts a query, the recipient has a task to answer this question.

Axiom: (effects of action *acceptQ*):

$$a = acceptQ(agt, query(e, agt, q)) \supset holds(has\_task(agt, e, q)), do(a, s)). \quad (6.4)$$

6.2.3 Action: $replyQ(e', query(e, e', q))$

The answer to a question may be either "*Yes*" or "*No*". "Yes" means that from the perspective of the requested agent the queried fluent can be proved to hold, which means that the fluent holds at the situation to reply; "No" means that from the perspective of the requested agent the queried fluent does not hold. In particular, for a query about

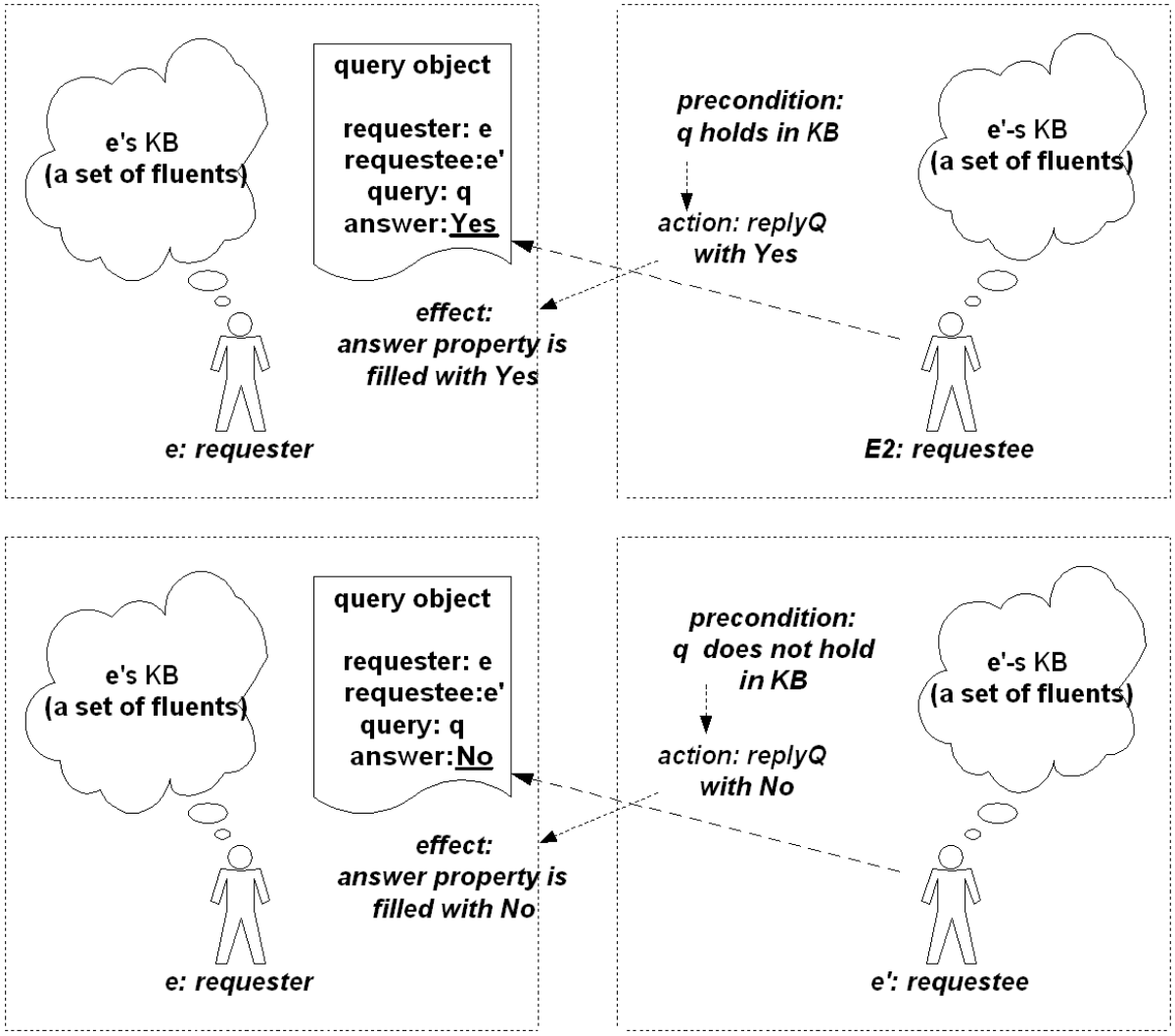

Figure 6.3: Action: replyQ

trust relationship, "No" means that there is no queried trust relationship, but it does not mean a distrust relationship. Action replyQ is illustrated in figure 6.3.

To answer a question with "Yes", the recipient must have a task to answer this question, and the questioned fluent q holds; to answer a question with "No", the recipient must have a task to answer this question; and fluent q does not hold.

Axiom (precondition of action $replyQ$):

$$Poss(replyQ(agt, query(e, agt, q), Yes), s) \equiv$$
$$holds(has\_task(agt, e, q), s) \wedge holds(q, s). \quad (6.5)$$

$$Poss(replyQ(agt, query(e, agt, q), No), s) \equiv$$
$$holds(has\_task(agt, e, q), s) \wedge \neg holds(q, s). \quad (6.6)$$

Action $replyQ(agt, query(e, agt, q), w)$ sets the "answer" property of object "query(e,agt,q)" as w, which is either "Yes" or "No"; after the requested agent replies the query, the requested agent has finished the task to answer the query, so that fluent $has\_task(agt, q)$ no longer holds.

After the requested agent executes action "*reply*" to a query, the query has an answer. In a implementation, this action sets the "answer" property of object "query(e,agt,q)" as "Yes" or "No" as replied; after the requested agent answers the query, the requested agent has finished the task to answer the query, so that fluent $has\_task(agt, q)$ no longer holds.

Axiom (effects of action $replyQ$):

$$a = replyQ(agt, query(e, agt, q), w) \supset \neg holds(has\_task(agt, e, q), do(a, s)). \quad (6.7)$$

6.2.4 Action: $checkAnswer(e, query(e, e', q), w)$

Action checkAnswer is illustrated in figure 6.4. The requester can check the answer of a query, if the query is there. In other words, the requester can check the answer at any situation after the query has been made. So action "*checkAnswer*" has the following precondition.

Axiom (precondition of action $checkAnswer$):

$$Poss(checkAnswer(e, query(e, e', q), w), s) \equiv holds(has\_query(query(e, e', q)), s). \quad (6.8)$$

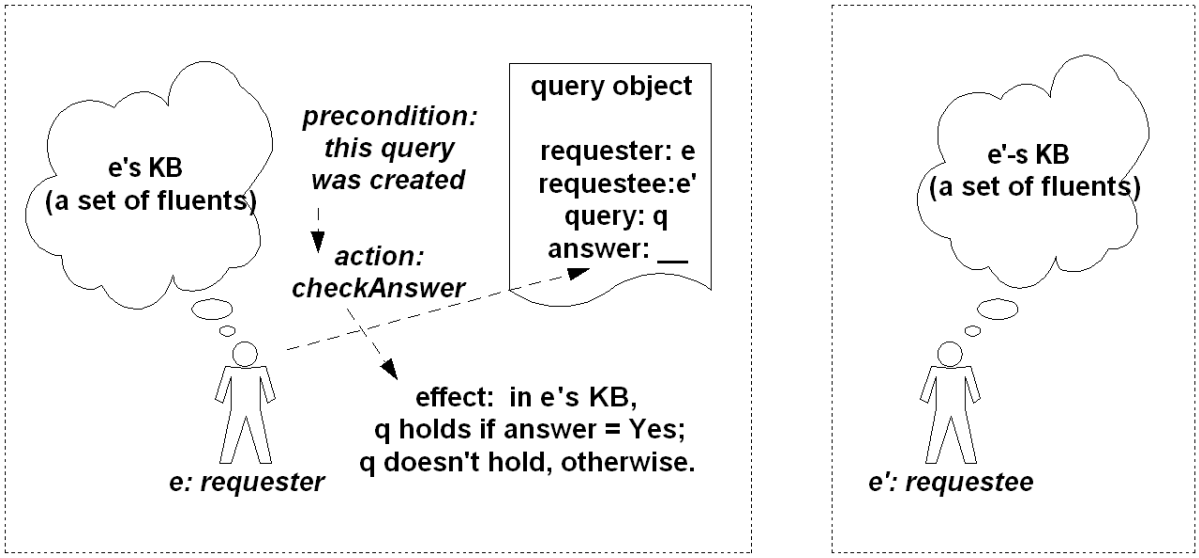

Figure 6.4: Action: checkAnswer

A query object has a property (a relational fluent) "$has\_answer(query(e, e', q), w)$", where w is the answer to the query. The default value of the answer (w) is "*Unknown*", which means no answer obtained from the requested agent.

If the requester checked the answer to a query, then the query has the answer as checked.

Axiom (effect of action $checkAnswer$**):**

$$a = checkAnswer(e, query(e, e', q), w) \supset holds(has\_answer(query(e, e', q), w), do(a, s)). \tag{6.9}$$

If the answer to a query is "*Yes*", then the queried fluent holds.

Axiom TR-8

$$holds(has\_answer(query(e, e', q), Yes), s)) \supset holds(q, s)). \tag{6.10}$$

When a query is made, queries among agents could quickly spread out in a social network. A problem is when such queries should stop propagation. A simple solution is that the questioning agent opens a web-accessible "signal" when making a query, and all following queries are associated with this signal; the condition for an agent to make

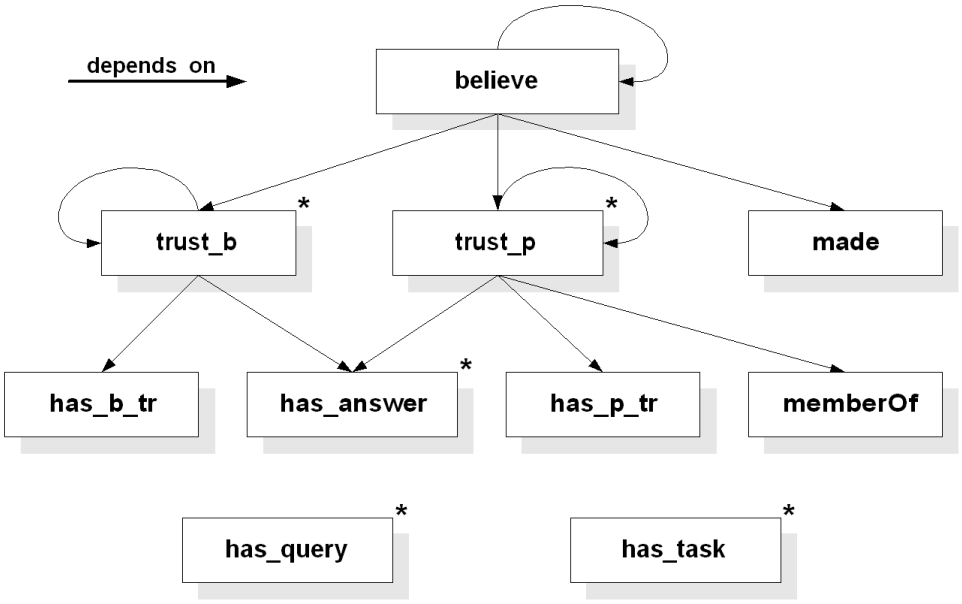

Figure 6.5: Dependency relation among trust related fluents

actions is that the signal is on; when the questioning agent receives an answer of "Yes" from any of questioned friends, the agent turn off the signal, which makes all actions associated with this signal stop. This type of signal also has a limited life length, that is to say, after a period of time, any efforts to find a trust path will stop. Since this is one of the technical issues regarding implementation, this paper will not include it in our formalization.

6.3 Successor State Axioms

This section discusses the successor state axioms of trust related fluents. First, we look at the dependency relation among the trust related fluents.

Figure 6.5 shows the dependency relation among the trust related fluents. The dependency relation is determined by the axioms and theorems introduced in the previous chapter. In the figure, the dependent fluents, which depend on other fluents, include: *believe*, *trust_b* and *trust_p*. The primary fluents, which do not depend on other fluents,

include *has_b_tr*, *has_p_tr*, *memberOf*, *made*, *has_query*, *has_answer*, and *has_task*.

In the figure, the fluents marked with * change with the actions discussed in previous section. From the effect axioms of those actions, fluents *has_query*, *has_answer*, and *has_task* will change with the actions. Furthermore, as stated in the previous section, in a query from one agent to another, the questioned fluent can be one of *trust_b* and *trust_p*; therefore, in addition to the dependency relation in figure 6.5, these two fluents may also be changed by the actions discussed in previous section.

In the following, we construct the successor state axioms for the fluents that directly change with actions. The changes of these fluents will cause the changes of other dependent fluents, which are calculated in accordance with the axioms and theorems presented in the previous chapter.

Fluent $has\_query(query(e, e', q))$ holds, if and only if: the agent just conducted the action of request, or the query has been made before the current action.

Successor State Axiom of fluent has_query**:**

$$holds(has\_query(query(e, e', q)), do(a, s)) \equiv$$
$$a = request(e, e', query(e, e', q))$$
$$\vee\ has\_query(query(e, e', q)), s). \quad (6.11)$$

Fluent $has\_task(e', q)$ holds, if and only if: the action just conducted is an action of accepting a query about question q; or the fluent originally holds and the action conducted is not to reply the question.

Successor State Axiom of fluent has_task**:**

$$holds(has\_task(e', q), do(a, s)) \equiv$$
$$a = acceptQ(e', query(e, e', q))$$
$$\vee\ holds(has\_task(e', q), s) \wedge a \neq replyQ(e', query(e, e', q), w). \quad (6.12)$$

Fluent $has\_answer(query(e, e', q), w)$ has following successor state axiom. Fluent

$has\_answer(query(e, e', q), w)$ holds, if and only if: the agent just conducted the action of "checkAnswer", and the answer is the same; or the fluent originally holds, and the action just done is not the action of "checkAnswer" with a different answer.

Successor State Axiom of fluent has_answer**:**

$$holds(has\_answer(query(e, e', q), w), do(a, s)) \equiv$$
$$a = checkAnswer(e, query(e, e', q), w)$$
$$\vee\, holds(has\_answer(query(e, e', q), w), s)$$
$$\wedge\, a \neq checkAnswer(e, query(e, e', q), w') \wedge w' \neq w. \quad (6.13)$$

As discussed earlier, if fluent $has\_answer(query(e, e', q), Yes)$ holds, the queried fluent q holds. This further causes other dependent fluents change.

Fluets *trust_b* and *trust_p* have successor state axioms with the following same semantic structure. Fluent f holds at situation *do(a,s)*, if and only if: the action just done (a) is to check the answer of the query regarding this fluent (f) and the answer is "Yes", or the fluent has already held in situation s.

Successor State Axiom of fluent $trust\_b$**:**

$$holds(trust\_b(e, e', x, k), do(a, s)) \equiv$$
$$a = checkAnswer(agt, query(agt, e, trust\_b(e, e', x, k)), w)$$
$$\wedge\, w = Yes$$
$$\vee\, holds(trust\_b(e, e', x, k), s). \quad (6.14)$$

Successor State Axiom of fluent $trust\_p$**:**

$$holds(trust\_p(e, e', x, k), do(a, s)) \equiv$$
$$a = checkAnswer(agt, query(agt, e, trust\_p(e, e', x, k)), w)$$
$$\wedge\, w = Yes$$
$$\vee\, holds(trust\_p(e, e', x, k), s). \quad (6.15)$$

6.4 Distributed Trust Reasoning: Example

We explain how the distributed trust system works by giving an example. For simplicity, this example only has a few of entities in social networks. In real applications, there could be a large number of entities involved, and a trust path may be long, although it is expected to be around 6 by the *six degree separation* law [131][43]. In addition, in order to focus on how agents interact each other to solve the trust judgment problem, we assume that all contexts are the same. In real applications, the contexts can be different, and they can be handled in trust reasoning as presented earlier.

The story is that entity E_1 needs to determine whether entity E_6 can be trusted regarding E_6's performance in context K, but E_1 does not know E_6 directly, so E_1 requests his trusted friends E_2, E_3 and E_4 (to whom E_1 has inter-individual trust in belief relationships in context of K) whether they trust E_6. The friends may further ask their friends about the same question. When E_1 gets answers from his friends, the trust relationship between E_1 and E_6 may be derived. The overall process is shown in figure 6.6.

In the following, we simulate the actions and the changes of situations in each entity's world in the following figures.

6.4.1 Changes in E_1's World

In E_1's world, at initial situation $S_0^{(1)}$, E_1 needs to determine whether the trust relationship $trust\_p(E_1, E_6, x, K)$ holds, but E_1 does not know who is E_6.

Initial Situation $S_0^{(1)}$ Assume, at initial situation $S_0^{(1)}$, the following fluents hold.

$$holds(has\_b\_tr(E_1, E_2, x, K), S_0^{(1)}). \tag{6.16}$$

$$holds(has\_b\_tr(E_1, E_3, x, K), S_0^{(1)}). \tag{6.17}$$

$$holds(has\_b\_tr(E_1, E_4, x, K), S_0^{(1)}). \tag{6.18}$$

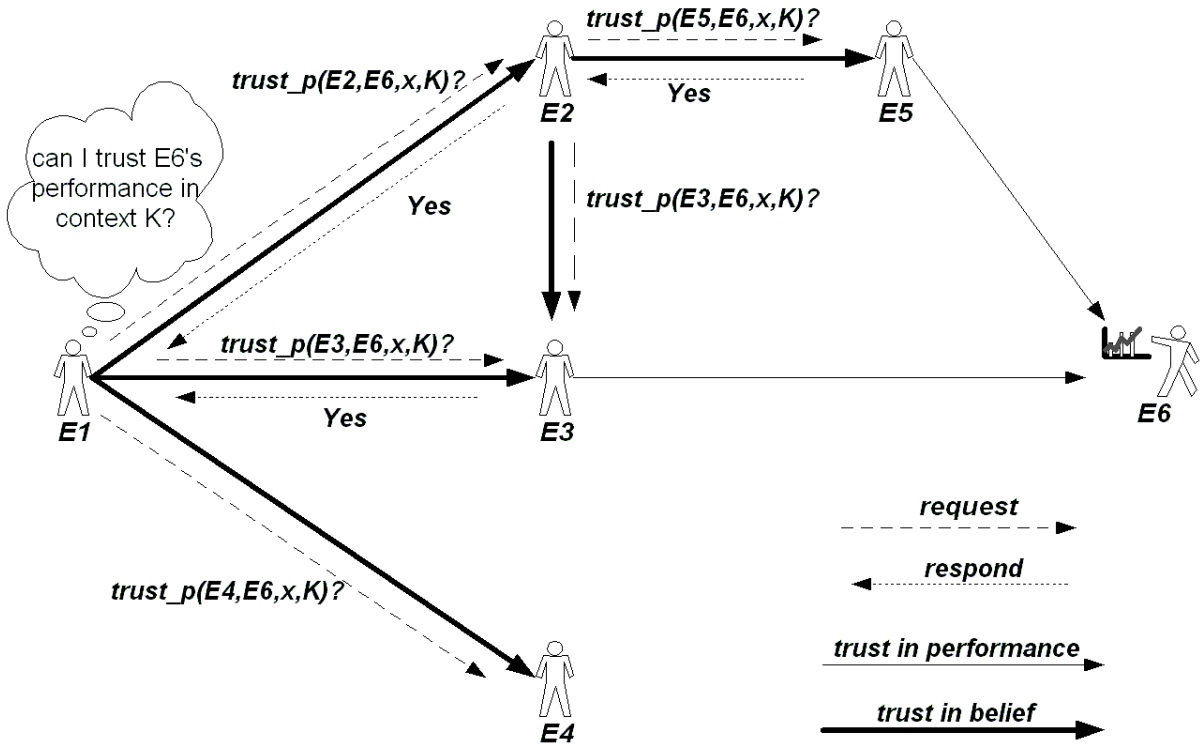

Figure 6.6: Example: distributed trust reasoning in social networks

Some other domain related fluents may also hold, but we ignore them here because they are irrelevant to the trust judgment.

Request Actions In order to infer whether $trust\_p(E_1, E_6, x, K)$ holds, E_1 requests his trusted friends E_2, E_3 and E_4 in context K.

$$S_1^{(1)} = do(request(E_1, query(E1, E_2, trust\_p(E_2, E_6, x, K))), S_0^{(1)}); \tag{6.19}$$

$$S_2^{(1)} = do(request(E_1, query(E1, E_3, trust\_p(E_3, E_6, x, K))), S_1^{(1)}); \tag{6.20}$$

$$S_3^{(1)} = do(request(E_1, query(E1, E_4, trust\_p(E_4, E_6, x, K))), S_2^{(1)}). \tag{6.21}$$

After these actions, queries to E_2, E_3 and E_4 are generated. By effect axiom of action *request*, formula (6.2),

$$holds(has\_query(query(E_1, E_2, trust\_p(E_2, E_6, x, K))), S_3^{(1)}); \tag{6.22}$$

$$holds(has\_query(query(E_1, E_3, trust\_p(E_3, E_6, x, K))), S_3^{(1)}); \tag{6.23}$$

$$holds(has\_query(query(E_1, E_4, trust\_p(E_4, E_6, x, K))), S_3^{(1)}). \tag{6.24}$$

Of course, E_1 can request his friends one by one until gets expected answer.

By successor state axiom of fluent *has_b_tr* (6.14), these request actions do not change these fluents. So the following fluents held at situation $S_3^{(1)}$

$$holds(has\_b\_tr(E_1, E_2, x, K), S_3^{(1)}); \tag{6.25}$$

$$holds(has\_b\_tr(E_1, E_3, x, K), S_3^{(1)}); \tag{6.26}$$

$$holds(has\_b\_tr(E_1, E_4, x, K), S_3^{(1)}); \tag{6.27}$$

$$holds(has\_query(query(E_1, E_2, trust\_p(E_2, E_6, x, K))), S_3^{(1)}); \tag{6.28}$$

$$holds(has\_query(query(E_1, E_3, trust\_p(E_3, E_6, x, K))), S_3^{(1)}); \tag{6.29}$$

$$holds(has\_query(query(E_1, E_4, trust\_p(E_4, E_6, x, K))), S_3^{(1)}). \tag{6.30}$$

Now we temporally turn to other entities' worlds and return E_1's world later.

6.4.2 Changes in E_2's World

In E_2's world, E_2 accepts the query from E_1; request his trusted friend E_3 but receives negative answer; so further request E_5.

Initial Situation $S_0^{(2)}$ At initial situation $S_0^{(2)}$, the following fluents hold.

$$holds(has\_b\_tr(E_2, E_3, x, K), S_0^{(2)}); \tag{6.31}$$

$$holds(has\_b\_tr(E_2, E_5, x, K), S_0^{(2)}). \tag{6.32}$$

Accept Query, to Situation $S_1^{(2)}$ Since no one ask E_2 about the question yet, we have

$$\neg holds(has\_task(E_2, \_, trust\_p(E_2, E_6, x, K)), S_0^{(2)}),$$

so, from the precondition axiom (6.3), E_2 can accept the query. Then,

$$S_1^{(2)} = do(acceptQ(E_2, query(E1, E_2, trust\_p(E_2, E_6, x, K))), S_0^{(2)}); \tag{6.33}$$

by effect axiom of this action (6.4), or successor state axiom (6.12),

$$holds(has\_task(E_2, E_1, trust\_p(E_2, E_6, x, K)), S_1^{(2)}); \tag{6.34}$$

by successor state axiom of fluent *has_b_tr*, formula (6.14), these fluents do not change with the action,

$$holds(has\_b\_tr(E_2, E_3, x, K), S_1^{(2)}) \tag{6.35}$$

$$holds(has\_b\_tr(E_2, E_5, x, K), S_1^{(2)}). \tag{6.36}$$

Request E_3, to Situation $S_3^{(2)}$ Right now, E_2 has a task to answer whether

$$trust\_p(E_2, E_6, x, K)$$

holds. Since E_2 does not know E_6 either, E_2 asks his trusted friend E_3 first.

$$S_2^{(2)} = do(request(E_2, query(E2, E_3, trust\_p(E_3, E_6, x, K))), S_1^{(2)}) \tag{6.37}$$

$$\tag{6.38}$$

By effect axiom of this action (6.2)

$$holds(has\_query(query(E_2, E_3, trust\_p(E_3, E_6, x, K))), S_2^{(2)}). \tag{6.39}$$

By successor state axiom (6.14) and (6.12), previous fluents do not change with the action. So,

$$holds(has\_b\_tr(E_2, E_3, x, K), S_2^{(2)}); \tag{6.40}$$

$$holds(has\_b\_tr(E_2, E_5, x, K), S_2^{(2)}); \tag{6.41}$$

$$holds(has\_task(E_2, E_1, trust\_p(E_2, E_6, x, K)), S_2^{(2)}). \tag{6.42}$$

By the precondition of action *checkAnswer* and fact (6.39), E_2 can check the answer of his query at or after situation $S_2^{(2)}$. Assume E_2 checks the answer at $S_2^{(2)}$.

$$S_3^{(2)} = do(checkAnswer(E_2, query(E_2, E_3, trust\_p(E_3, E_6, x, K)), w), S_2^{(2)}). \quad (6.43)$$

By successor state axiom of fluent *has_answer* (6.13),

$$holds(has\_answer(query(E2, E_3, trust\_p(E_3, E_6, x, K)), w), S_3^{(2)}). \quad (6.44)$$

Assume that for some reasons, the answer obtained is "*Unknown*", i.e. $w = Unknown$. By the state constraints, this answer does not change any dependent fluents.

By successor state axioms (6.14), (6.12), (6.11) and (6.13), previous fluents still hold. So, the following fluents hold at situation $S_3^{(2)}$:

$$holds(has\_b\_tr(E_2, E_3, x, K), S_3^{(2)}); \quad (6.45)$$

$$holds(has\_b\_tr(E_2, E_5, x, K), S_3^{(2)}); \quad (6.46)$$

$$holds(has\_task(E_2, E_1, trust\_p(E_2, E_6, x, K)), S_3^{(2)}); \quad (6.47)$$

$$holds(has\_query(query(E_2, E_3, trust\_p(E_3, E_6, x, K))), S_3^{(2)}); \quad (6.48)$$

$$holds(has\_answer(query(E2, E_3, trust\_p(E_3, E_6, x, K)), w), S_3^{(2)}). \quad (6.49)$$

Request E_5, to Situation $S_4^{(2)}$ Since the queried problem remains unsolved, i.e. fluent

$$holds(has\_task(E_2, E_1, trust\_p(E_2, E_6, x, K)), S_3^{(2)})$$

still holds at current situation, E_2 further requests his another trusted friend E_5.

$$S_4^{(2)} = do(request(E_2, query(E2, E_5, trust\_p(E_5, E_6, x, K))), S_3^{(2)}). \quad (6.50)$$

By effect axiom of this action (6.2)

$$holds(has\_query(query(E_2, E_5, trust\_p(E_5, E_6, x, K))), S_4^{(2)}). \quad (6.51)$$

By successor state axiom (6.14) and (6.12), all previous fluents do not change with the action. So, the following fluents hold at situation $S_4^{(2)}$:

$$holds(has\_b\_tr(E_2, E_3, x, K), S_4^{(2)}); \tag{6.52}$$

$$holds(has\_b\_tr(E_2, E_5, x, K), S_4^{(2)}); \tag{6.53}$$

$$holds(has\_task(E_2, E_1, trust\_p(E_2, E_6, x, K)), S_4^{(2)}); \tag{6.54}$$

$$holds(has\_query(query(E_2, E_3, trust\_p(E_3, E_6, x, K))), S_4^{(2)}); \tag{6.55}$$

$$holds(has\_answer(query(E2, E_3, trust\_p(E_3, E_6, x, K)), w), S_4^{(2)}) \tag{6.56}$$

$$holds(has\_query(query(E_2, E_5, trust\_p(E_5, E_6, x, K))), S_4^{(2)}). \tag{6.57}$$

Now we temporally turn to other entities' worlds and return E_2's world later.

6.4.3 Changes in E_3's World

E_3 accepts E_1's query, but reject query from E_2 for E_3 being in busy status to answer the same question; E_3 solves the queried question and replies E_1's query with "*Yes*".

Initial Situation $S_0^{(3)}$ In E_3's world, initially,

$$holds(has\_p\_tr(E_3, E_6, x, K), S_0^{(3)}). \tag{6.58}$$

Accept Query, to Situation $S_1^{(3)}$ E_3 receives a query from E_1 at situation $S_0^{(3)}$. Since no question is received yet,

$$\neg holds(has\_task(E_3, \_, trust\_p(E_3, E_6, x, K))), S_0^{(3)}),$$

E_3 accepts the query, i.e.

$$S_1^{(3)} = do(acceptQ(E_3, query(E1, E_3, trust\_p(E_3, E_6, x, K))), S_0^{(3)}); \tag{6.59}$$

then by effect axiom of this action (6.4),

$$holds(has\_task(E_3, E_1, trust\_p(E_3, E_6, x, K)), S_1^{(3)}); \tag{6.60}$$

by successor axiom of fluent *has_b_tr* (6.14), this type of fluent remains unchanged; so we have fluents held at $S_1^{(3)}$,

$$holds(has\_p\_tr(E_3, E_6, x, K), S_1^{(3)}); \tag{6.61}$$

$$holds(has\_task(E_3, E_1, trust\_p(E_3, E_6, x, K)), S_1^{(3)}). \tag{6.62}$$

If at this situation, E_3 receives query from E_2 about the same question, because fact (6.62), i.e. E_3 has already had task to answer the same question, by the precondition axiom of action *acceptQ* (6.3), E_3 is no longer be able to accept this query. So E_2's query $query(E2, E_3, trust\_p(E_3, E_6, x, K))$ will keep having the default answer "*Unknown*".

Solve Queried Problem E_3 solves the queried problem by reasoning as follows. From theorem TR-3 (C.19) and fact (6.61),

$$holds(trust\_p(E_3, E_6, x, K), S_1^{(3)}). \tag{6.63}$$

Reply Query, to Situation $S_2^{(3)}$ Now, from the precondition of action *replyQ*, E_3 can reply the query with "*Yes*", i.e.

$$S_2^{(3)} = do(replyQ(E_3, query(E1, E_3, trust\_p(E_3, E_6, x, K)), Yes), S_1^{(3)}). \tag{6.64}$$

By successor state axioms of *has_task*, fluent

$$has\_task(E_3, trust\_p(E_3, E_6, x, K))$$

no longer holds at situation $S_2^{(3)}$. So, if E_3 receives query from E_2 at this time, E_3 will accept the query.

6.4.4 Entity E_4

Similar to entities E_2 and E_3, E_4 can accepts the query from E_1. However, for some reasons regarding privacy, security or other social factors, E_4 does not feel like answering

this question. Therefore, if E_1 checks the answer for this query, the answer is always "*Unknown*".

Consider another possibility. E_4 does not know E_6 and queried his friends, but failed to receive an answer of "*Yes*" in a number of request actions, so he replies with "*No*". The actions of *acceptQ* and *replyQ*, and the changes of fluents are similar, so the details are omitted.

6.4.5 Changes in E_5's World

E_5 accepts the query from E_2, solves the queried problem, then replies to the query.

Initial Situation $S_0^{(5)}$ In E_5's world, assume E_5 has inter-individual trust relationship with E_6 at the initial situation $S_0^{(5)}$,

$$holds(has\_p\_tr(E_5, E_6, x, K), S_0^{(5)}). \tag{6.65}$$

Accept Query First, E_5 accepts the query from E_2,

$$S_1^{(5)} = do(acceptQ(E_5, query(E2, E_5, trust\_p(E_5, E_6, x, K))), S_0^{(5)}); \tag{6.66}$$

then by successor state axioms (6.12) and (6.14), at situation $S_1^{(5)}$,

$$holds(has\_task(E_5, E_2, trust\_p(E_5, E_6, x, K)), S_1^{(5)}), \tag{6.67}$$

$$holds(has\_p\_tr(E_5, E_6, x, K), S_1^{(5)}). \tag{6.68}$$

Solve Queried Problem Then, E_5 solves the queried problem by reasoning as follows.

By fact (6.68) and theorem TR-3 (C.19),

$$holds(trust\_p(E_5, E_6, x, K), S_1^{(5)});$$

Reply Query By precondition of action $replyQ$, E_5 can reply to E_2's query with "Yes",

$$S_2^{(5)} = do(replyQ(E_5, query(E_2, E_5, trust\_p(E_5, E_6, x, K)), Yes), S_1^{(5)}); \tag{6.69}$$

by successor state axioms (6.12), (6.14), after the $replyQ$ action, fluent

$$has\_task(E_5, E_2, trust\_p(E_5, E_6, x, K))$$

no longer holds at situation $S_2^{(5)}$.

6.4.6 Changes in E_2's World (2)

Return to E_2's world. Now, entity E_2 checks the answer of his query, solve his queried problem, and replies to the query from e_1.

Check Answer By the precondition of action $checkAnswer$, E_2 can check the answers of his query made earlier.

$$S_5^{(2)} = do(checkAnswer(E_2, query(E2, E_5, trust\_p(E_5, E_6, x, K)), w), S_4^{(2)}). \tag{6.70}$$

By effect axiom of this action (6.9),

$$holds(has\_answer(query(E_2, E_5, trust\_p(E_5, E_6, x, K)), w), S_5^{(2)}). \tag{6.71}$$

Since the reply by E_5 is "Yes", the answer obtained is "Yes", i.e. $w = Yes$. By axiom TR-8 (6.10),

$$holds(trust\_p(E_5, E_6, x, K), S_5^{(2)}). \tag{6.72}$$

By successor state axioms of primary fluents and state constraints for dependent

fluents, at current situation,

$$holds(has\_b\_tr(E_2, E_3, x, K), S_5^{(2)}); \tag{6.73}$$

$$holds(has\_b\_tr(E_2, E_5, x, K), S_5^{(2)}); \tag{6.74}$$

$$holds(has\_task(E_2, E_1, trust\_p(E_2, E_6, x, K)), S_5^{(2)}); \tag{6.75}$$

$$holds(has\_query(query(E_2, E_3, trust\_p(E_3, E_6, x, K))), S_5^{(2)}); \tag{6.76}$$

$$holds(has\_answer(query(E2, E_3, trust\_p(E_3, E_6, x, K)), Unknown), S_5^{(2)}) \tag{6.77}$$

$$holds(has\_query(query(E_2, E_5, trust\_p(E_5, E_6, x, K))), S_5^{(2)}); \tag{6.78}$$

$$holds(has\_answer(query(E_2, E_5, trust\_p(E_5, E_6, x, K)), Yes), S_5^{(2)}); \tag{6.79}$$

$$holds(trust\_p(E_5, E_6, x, K), S_5^{(2)}). \tag{6.80}$$

Solve Queried Problem E_2 solves the queried problem by reasoning as follows.

by theorem TR-3 and fact (6.89),

$$holds(trust\_b(E_2, E_5, x, K), S_5^{(2)}); \tag{6.81}$$

by theorem TR-8(a)(5.8) as well as facts (6.89) and (6.80), E_2 gets solution,

$$holds(trust\_p(E_2, E_6, x, K), S_5^{(2)}). \tag{6.82}$$

Reply Query By precondition of action $replyQ$, now E_2 can reply to E_1's query with "Yes",

$$S_6^{(2)} = do(replyQ(E_2, query(E1, E_2, trust\_p(E_2, E_6, x, K)), Yes), S_5^{(2)}); \tag{6.83}$$

by effect axiom of this action (6.7), after the $replyQ$ action, fluent

$$has\_task(E_2, E_1, trust\_p(E_2, E_6, x, K))$$

no longer holds at situation $S_6^{(2)}$.

By successor state axiom (6.14) and trust reasoning, the following fluents hold,

$$holds(has\_b\_tr(E_2, E_3, x, K), S_6^{(2)}); \tag{6.84}$$

$$holds(has\_b\_tr(E_2, E_5, x, K), S_6^{(2)}); \tag{6.85}$$

$$holds(trust\_p(E_5, E_6, x, K), S_6^{(2)}); \tag{6.86}$$

$$holds(trust\_p(E_2, E_6, x, K), S_6^{(2)}). \tag{6.87}$$

After the *replyQ* action, applying successor state axioms and state constraints for dependent fluents, the following fluents hold at current situation,

$$holds(has\_b\_tr(E_2, E_3, x, K), S_6^{(2)}); \tag{6.88}$$

$$holds(has\_b\_tr(E_2, E_5, x, K), S_6^{(2)}); \tag{6.89}$$

$$holds(has\_query(query(E_2, E_3, trust\_p(E_3, E_6, x, K))), S_6^{(2)}); \tag{6.90}$$

$$holds(has\_answer(query(E2, E_3, trust\_p(E_3, E_6, x, K)), Unknown), {}_6^{(2)}) \tag{6.91}$$

$$holds(has\_query(query(E_2, E_5, trust\_p(E_5, E_6, x, K))), S_6^{(2)}); \tag{6.92}$$

$$holds(has\_answer(query(E_2, E_5, trust\_p(E_5, E_6, x, K)), Yes), S_6^{(2)}); \tag{6.93}$$

$$holds(trust\_p(E_5, E_6, x, K), S_6^{(2)}); \tag{6.94}$$

$$holds(trust\_p(E_2, E_6, x, K), S_6^{(2)}). \tag{6.95}$$

6.4.7 Changes in E_1's World (2)

Return to E_1's world. Now, entity E_1 checks the answers of his queries and solve his problem.

Check Answer By the precondition of action *checkAnswer*, E_1 can check the answers of his queries made earlier. Assume E_1 checks the answer of the query sent to E_2 first.

$$S_4^{(1)} = do(checkAnswer(E_1, query(E1, E_2, trust\_p(E_2, E_6, x, K)), w), S_3^{(1)}). \tag{6.96}$$

Since the reply by E_2 is "Yes", the answer obtained is "Yes", i.e. $w = Yes$. By the effect axiom of the action (6.9),

$$holds(has\_answer(query(E_1, E_2, trust\_p(E_2, E_6, x, K)), Yes), S_4^{(1)}); \tag{6.97}$$

by axiom TY-8 (6.10),

$$holds(trust\_p(E_2, E_6, x, K), S_4^{(1)}). \tag{6.98}$$

By successor state axioms and state constraints for dependent fluents, in the current situation,

$$holds(has\_b\_tr(E_1, E_2, x, K), S_4^{(1)}); \tag{6.99}$$

$$holds(has\_b\_tr(E_1, E_3, x, K), S_4^{(1)}); \tag{6.100}$$

$$holds(has\_b\_tr(E_1, E_4, x, K), S_4^{(1)}); \tag{6.101}$$

$$holds(has\_query(query(E_1, E_2, trust\_p(E_2, E_6, x, K))), S_4^{(1)}); \tag{6.102}$$

$$holds(has\_query(query(E_1, E_3, trust\_p(E_3, E_6, x, K))), S_4^{(1)}); \tag{6.103}$$

$$holds(has\_query(query(E_1, E_4, trust\_p(E_4, E_6, x, K))), S_4^{(1)}); \tag{6.104}$$

$$holds(has\_answer(query(E_1, E_2, trust\_p(E_2, E_6, x, K)), Yes), S_4^{(1)}); \tag{6.105}$$

$$holds(trust\_p(E_2, E_6, x, K), S_4^{(1)}). \tag{6.106}$$

Solve Queried Problem E_1 solves his problem by reasoning as follows. By theorem TR-3 and fact (6.99)

$$holds(trust\_b(E_1, E_2, x, K), S_4^{(1)}); \tag{6.107}$$

by theorem TR-8(a)(5.8) as well as facts (6.107) and (6.106), E_1 gets solution,

$$holds(trust\_p(E_1, E_6, x, K), S_4^{(1)}). \tag{6.108}$$

In this way, E_1 solved his problem in a distributed computing manner.

6.5 Summary

Most proposed social networks based trust models require to access to all personal trust data of the visited entities in social networks in a search. However, for the reason of privacy, people usually do not publish their private data to the public online. For this problem, this paper proposed a social network-based distributed trust reasoning model and constructed the model in situation calculus, which facilitates social network-based trust judgment. In real applications, the model can be implemented with web services, then each entity in social networks only answers whether to trust the questioned entity, and the private trust data need not to be revealed to public. In addition, by this distributed model, a search for a trust path is virtually carried out in parallel in a social network, so that the shortest trust path could be found in a short time with high probability.

Chapter 7

KP Application in Finance

In this chapter, in order to demonstrate the potential uses of knowledge provenance, we develop a KP application case in the field of financial investment.

Financial investment is a typical field where people need a great amount of information to make decisions. Particularly, with the advent of the Web age, investors are flooded with a great variety of web information. However, as discussed in Chapter 1, information validity is always a problem in the area. Knowledge provenance can help to increase financial information trustworthiness.

7.1 Financial Information and XBRL

Financial statements are essential information for the capital market. In the market, companies are required to provide their financial statements to market regulators such as U.S. Securities and Exchange Commission (SEC). Market regulators collect and disseminate companies' financial information to investors. Professional financial analysts compile and interpret financial data and other relevant information and analyze the current situation and future trends of the markets in various financial analysis reports, in particular, investment newsletters. These reports help investors make their investment decisions.

However, a surprising fact about the validity of financial data is revealed by Christopher Cox, Chairman of U.S. Securities and Exchange Commission, in a speech on 3 March 2006. Cox addressed that the error rate of the data used by financial analysts in their valuation models is up to 28% or higher [32]. The direct reason for this problem is that the financial data used in financial analysis come from neither the companies nor SEC, instead the data is bought from intermediate companies that manually re-key financial statements into the data of suitable formats. Another reason from a technical view is that the financial data is mixed with text in financial statements, so that the data cannot be extracted automatically by machines. This situation yields the needs of "interactive data"[32] – machine-processable data that can be easily extracted from financial statements and customized into the reports that meet the different needs of financial information users.

To solve the problem, XBRL [189](e**X**tensible **B**usiness **R**eporting **L**anguage) has been developed to facilitate electronic communication of business and financial data. Many countries have initiated or suggested the use of XBRL. For examples, US SEC set up an "interactive-data test group" in which companies are using XBRL to make their SEC fillings [32]; UK Government proposes to make the use of XBRL mandatory for company tax filing from March 2010 [25]. The use of XBRL will make the data used in financial analysis more accurate and reliable.

Financial analysis reports, as well as financial news stories, usually contain many data from companies' financial statements. The use of XBRL makes it possible to validate these data by machines. In section 7.2, under the framework of KP, we discuss the approach to annotating and validating XBRL data in web based financial reports; in section 7.3, we discuss how to use KP for finance information / knowledge provenance reasoning.

Before finishing this section, we briefly introduce the structure of XBRL files as follows.

7.1.1 XBRL Data Structure

Business information reported by XBRL is specified in two types of documents: instances and taxonomies. An XBRL instance file is an XML file that contains the business data being reported; an XBRL taxonomy file is an XML Schema file that defines the structure of XBRL instances. Each item of business data is specified as an XML element, called "item element". The tag of an item element is the name of the data item; the content of an item element is the value of the data item; and each item element also has some other attributes such as the unit and the context. The context of an item refers to the entity (company id) and the period (startdate and enddate) associated with the data item.

Example. In a Microsoft's financial statement in XBRL, the revenue of the fourth quart of 2005 is specified as the following item element.

$< usfr - pte : OperatingRevenue$

$contextRef =$ "$P3MQ2FY2006''$

$decimals =$ "$-6''$

$unitRef =$ "$USD'' >$

11837000000

$< /usfr - pte : OperatingRevenue >$

The context of this item is defined by the following context element.

$< xbrli : context\ rdf : id =$ "$P3MQ2FY2006'' >$

$< xbrli : entity >$

$< xbrli : identifier$

$scheme =$ "$http : //www.sec.gov/CIK'' >$

0000789019

$< /xbrli : identifier >$

$< /xbrli : entity >$

$< xbrli : period >$

$< xbrli : startDate > 2005 - 10 - 01 < /xbrli : startDate >$

$< xbrli : endDate > 2005 - 12 - 31 < /xbrli : endDate >$

$< /xbrli : period >$

$< /xbrli : context >$

The unit of this item is defined by the following unit element.

$< xbrli : unitid = "USD" >$

$< xbrli : measure > iso4217 : USD < /xbrli : measure >$

$< /xbrli : unit >$

7.2 Financial Data Annotation and Validation

From the view of KP, each data item used in financial analysis reports (including news stories) is a proposition. In order to conduct provenance reasoning, each data item needs to be annotated with KP tags. Corresponding to this, each data item in XBRL files also needs to be annotated. However, annotating each data item in XBRL files is a very inefficient solution.

In this section, we present a special solution for XBRL data annotation and validation. Each item of data in XBRL files is not annotated, instead each data item is annotated only in its use in financial reports; an XBRL data item has "*believed truth value*" of "*true*" if it is contained in a valid XBRL instance file.

7.2.1 Annotation

Consider the following news story: "Microsoft Corp. (Nasdaq: MSFT - News) today announced revenue of $11.84 billion (**11837000000**) for the quarter ended December 31, 2005, a 9% increase over the same period of the prior year, marking the highest quarterly revenue in the company's history." In the news, **11837000000** is an item of data in

Microsoft's financial statement reported in XBRL. How should this item of XBRL data be annotated with kp tags in this financial news report?

In order to annotate the data items, which come from XBRL files, in financial reports, we define *XBRL_DataItem* as a subclass of *Asserted_prop*.

Now, we illustrate how to annotate XBRL data in financial reports by giving an example. In the above story, **11837000000** is an item of data from a XBRL file. This string can be annotated with KP tags in the news report as follows.

<html>...

<body>...

Microsoft Corp. (Nasdaq: MSFT - News) today announced

<kp-br:XBRL_DataItem rdf:id = "#MSFT-Q-revenue-20051231"

kp:inXBRLDoc="http://www.sec.gov/Archives/edgar/data/

789019/000119312506050847/xmsft-20051231.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:OperatingRevenue"

kp-br:value = "10818000000"

kp-br:unit = "USD"

kp-br:startDate = 2005-10-01

kp-br:endDate = 2005-12-31

kp-br:entity = "Microsoft">

revenue of $11.84 billion for the quarter ended December 31, 2005,

</kp-br:XBRL_DataItem>

a 9% increase over the same period of the prior year, marking the highest quarterly revenue in the company's history.

...

</body>

</html>

7.2.2 Authenticity Validation of XBRL Data

From the discussion above, we have seen that a financial report may contain some data items that come from companies's financial statements in XBRL. In this subsection, we discuss how KP reasoner validates the authenticity of those XBRL data items contained in financial reports.

KP reasoner validates XBRL data items appeared in financial reports by making a XQuery[190]. We present this method by continuing our the previous example, as shown as follows:

(1) get the XBRL instance file containing the data in question. The url of the XBRL file is specified in the *inDoc* attribute of *XBRL_Item*. In this example, the url is:
http://www.sec.gov/Archives/edgar/data/789019/000119312506050847
/xmsft-20051231.xml;

(2) make a XQuery from the XBRL instance file to get item elements with tag "usfr-pte:OperatingRevenue". The query is:

doc(xmsft-20051231.xml)/xbrl/usfr-pte:OperatingRevenue.

doc() is the function that is used by XQuery to extract elements from an XML document. XQuery will return all elements with the tag.

(3) check whether the questioned XBRL data item matches any XBRL element extracted by XQuery . The attributes need to be validated include: value, the unit and the context (entity and the period of the item). Validating context requires to make another XQuery to extract the context element by "contextRef" attribute. Validating entity needs to check the entity's identification number, typically, *CIK* issued by SEC.

Note that it is the information creator's responsibility to make the annotated text have the same semantics as the metadata of the XBRL_DataItem has; it is the information users' responsibility to check the text and the metadata have the same context. KP reasoner only validate whether the metadata match the item in the specified XBRL instannce file.

After a XBRL data item in question is successfully validated, predicate

$$has\_authentic\_source(x, c)$$

(defined in Chapter 3) becomes true, then, KP reasoner uses the axioms and theories developed in KP to make provenance reasoning.

7.3 Financial Knowledge Provenance

The use of XBRL will make the financial data used in financial analysis reports more accurate and reliable. However, XBRL alone cannot completely solve the validity problem in financial analysis. In financial analysis reports, there may be not only companies' financial data but also many other information from various information sources, many personal interpretations about data, personal opinions, personal assumptions, assertions and derived results. These features are beyond the problem XBRL targets.

To help investors judge the the quality of financial reports, tools have appeared to evaluate the performance of investment newsletters and financial advisors. These tools help investors judge the trustworthiness of financial analysis product brands and financial analysts based on their overall performance history. From the perspective of KP, these tools can help investors to evaluate their trust in these financial information sources.

In real financial investment situations, a specific financial proposition (such as an investment advice or opinion) usually is based on a number of facts or opinions from different information sources. The trustworthiness of this proposition depends not only on the proposition creator but also the dependency on other information as well as other information sources. Knowledge provenance can be used to evaluate the validity of this type of information, based on the degrees of trust in information sources. In this section, we introduce KP application in web based financial investment analysis.

The Web is a very convenient tool for both investors and financial companies. Many websites provide financial information services. Marketwatch.com is one of the typical

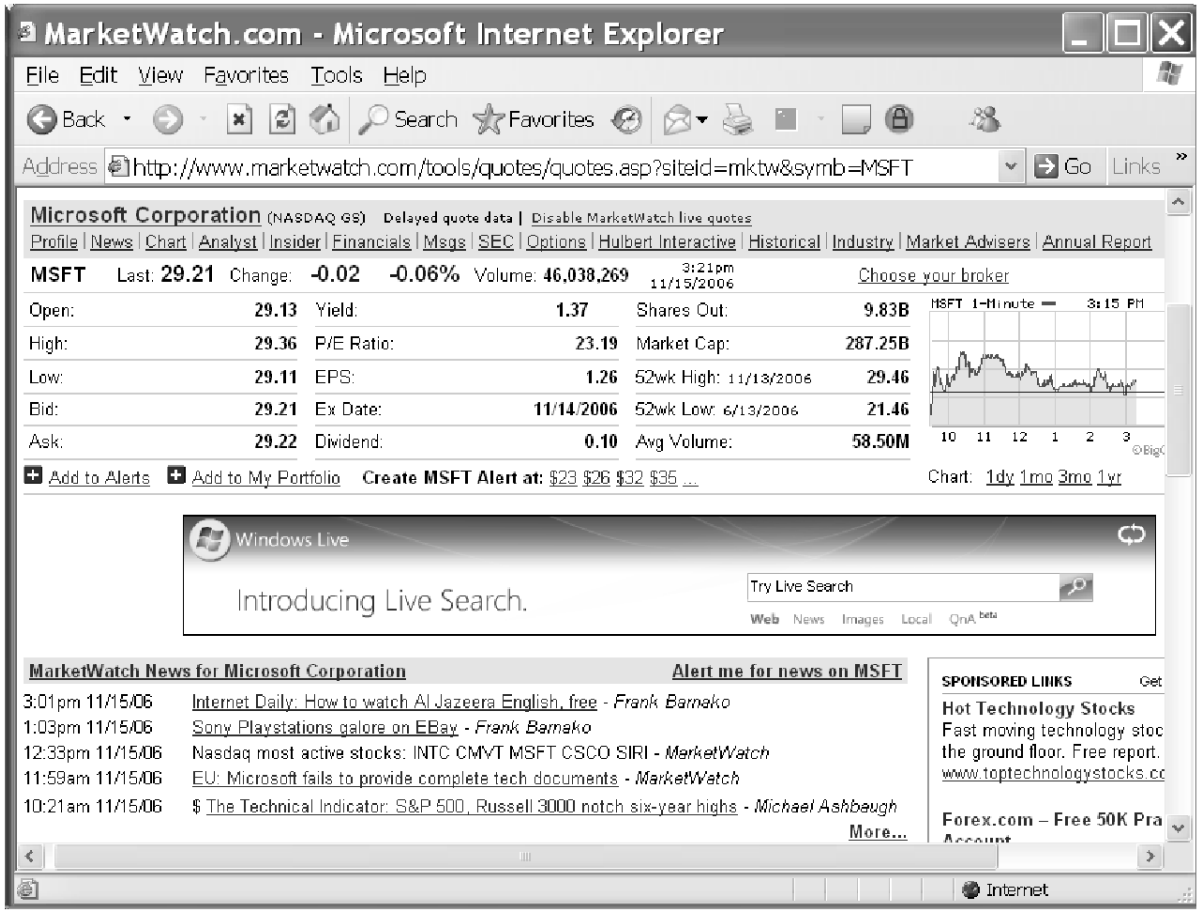

Figure 7.1: A snapshot from MarketWatch

examples.

MarketWatch provides tools for investors to study the market. Figure 7.1 shows a snapshot of MarketWatch for Microsoft. This page can be reached at:

http://www.marketwatch.com/tools/quotes

/quotes.asp?siteid=mktw&symb=MSFT

For a given company in the market, MarketWatch integrates instant market data (automatically updated), news, analysis reports, investment newsletter rating, the company's SEC fillings and annual reports. MarketWatch is a useful tool for investors. However, similar to other websites of the same type, MarketWatch does not provide the provenance of financial information. For example, in a piece of news, it reports that

"Lockheed Martin upped to buy at Goldman Sachs", but no any provenance about the news is given. The use of KP could make the Web based financial information service more trustworthy.

In the following, In order to demonstrate how to use knowledge provenance and XBRL to solve financial information validity problem, we give a simplified example of investment newsletter as follows [1].

7.3.1 Example of Investment Newsletter

Issued by: W-finance.com

Issued Date: 15 November 2006

Company: Lockheed Martin Corporation (LMT-NYSE)

Recommendation : Buy

---

[1]This sample is adapted from the following materials:
(1) Zacks Digest on Lockheed Martin Corp., October 26, 2006.
(2) Lockheed Locks In Q3: Fool by Numbers
http://www.fool.com/News/mft/2006/mft06102549.htm (visited on 15 Nov. 2006).
(3) MarketWatch: Most Recent Analyst Recommendations on LMT
http://www.marketwatch.com/tools/quotes/snapshot.asp?symb=LMT (visited on 15 Nov. 2006).
(4) Lockheed Martin's profit jumps 47%
http://www.marketwatch.com/news/story/lockheed-martin-profit-climbs-47/
story.aspx?guid=%7BE010EEFE%2DAB64%2D4379%2DBD41%2DB5AC9C58FD65%7D (visited on 15 Nov. 2006).
(5) Lockheed to spearhead next push into space
http://www.marketwatch.com/News/Story/Story.aspx?guid={31F2F468-3705-4273-879E-94D9351924FD} (visited on 15 Nov. 2006).
(6) Rumsfeld's exit clouds defense policy
http://www.marketwatch.com/news/story/rumsfeld-departure-clouds-defense-policy/
story.aspx?guid=%7B205C5294%2DA6BE%2D44A2%2D9B32%2DE6E9A9303AFC%7D (visited on 15 Nov. 2006)
(7) Lockheed Martin's Financial Statement on Third Quarter of 2006 (XBRL files)
http://www.sec.gov/Archives/edgar/data/936468/000119312506219311/lmt-20060930.xml
http://www.sec.gov/Archives/edgar/data/936468/000119312506219311/lmt-20060930.xsd (visited on 15 Nov. 2006)

Target Price: $94.13

Current Price (2006-11-15): 88.54

52wk High (2006-10-24): 89.89

52wk Low (2005-11-15): 59.55

Key Positive Arguments :

- Expansion of margins;
- Strong free cash flow and improving balance sheet;
- Big increase of profits and reduced tax rate;
- Won major contracts, for example, Orion project;

Key Negative Arguments :

- Uncertainty in defense policy.

Recent Events :

- On November 8, 2006, Defense Secretary Donald H. Rumsfeld stepped down, which clouds defense policy.
- Democrats wins 2006 Elections.
- On October 24, 2006, LMT released Q3 earnings.
- On August 31, 2006, NASA selected Thursday Lockheed Martin Corp. as the prime contractor to design, develop, and build Orion, America's spacecraft for a new generation of explorers. The estimated value for the project is $3.9 billion.

Revenue Highlights :

Third-quarter net income rose 47% to $629 million, or $1.46 a share, from $427 million, or 96 cents a share, in the year-earlier period. Its tax rate dropped to 22.8% in the quarter from 30.3% a year ago.

Margins Highlights :

Gross margin [2] was up 1.67 point to 8.36% from 6.69% in the year-earlier period; operating margin [3] was up 1.71 to 9.42% from 7.67%.

Cash Flows Highlight :

Cash from operations (year to date) rose 9.9% to $3,450 million from $3,138 million.

Balance Sheet Highlights :

The major items of LMT balance sheet.

Balance Sheet (in millions	**Q3 2006**	**Q4 2005**	**Change**
Total assets	$29,093	$27,744	4.86%
Total current liabilities	$10,383	$9,428	10.13%
Stockholders equity	$8,083	$7,867	2.75%

7.3.2 Annotation

Assume the above investment newsletter is published in HTML. Appendix D demonstrates how to annotate financial reports with KP metadata by giving the KP annotation for this investment newsletter. In order to focus on KP, we neglect most xhtml tags and only give KP metadata.

[2] gross margin = (net sale - cost of sale) / net sale.
[3] operating margin = operating income / net sale.

7.3.3 Provenance Reasoning

This subsection demonstrates how to make provenance reasoning for the propositions in the sample investment newsletter.

When an investor, "John", reads the newsletter, he is interested in the message about Lockheed Martin, so he makes provenance request regarding the proposition "RecommToBuy" by using a KP reasoner; then KP reasoner makes provenance reasoning and answers John to what an extent this proposition can be believed to be true.

First, we assume John has the following trust relationships, which is inferred from John and his trusted friends' inter-individual trust relationships [4].

$$trusted\_in(John, W, \text{“}investment\ advice'').$$

$$trusted\_in(John, W, \text{“}finance\ analysis'').$$

$$trusted\_in(John, W, \text{“}news\ \&comments'').$$

$$trusted\_in(John, SEC, \text{“}finance\ statement'').$$

$$trusted\_in(John, NASA, \text{“}news\ in\ NASA'').$$

$$trusted\_in(John, CNN, \text{“}news\ in\ US\ Politics'').$$

The information dependencies among the propositions in the newsletter is illustrated in figure 7.2.

In provenance reasoning, KP reasoner handles a derived proposition with several supporting propositions as the derived proposition has one implicit support proposition and this implicit proposition is the conjunction of those support propositions.

The provenance reasoning process is given as the following steps, which are illustrated in figure 7.3.

(1) Provenance reasoning on derived proposition "RecommToBuy". This derived proposition is authored by W whom provenance requester (John) trusts in the

[4] In order to foucs on knowledge provenance, we omit how these trust relationships are derived from a social network.

field "investment advices"; assume that authentication is successful, by theorem kp-1, this proposition is believed. Since this proposition is dependent on several other propositions, by theorem KP-3, to determine the believed truth value of this derived proposition, KP reasoner needs to infer the believed truth values of those support propositions first.

(2) Provenance reasoning on derived proposition "Argument-Posi-1". As shown in figure 7.3, this derived proposition is authored by W whom provenance requester (John) trusts in the field "financial analysis"; assume that the authentication of this proposition is successful, by theorem kp-1, this proposition is believed; Since this proposition is further dependent on six items of data in LMT's financial statement in an XBRL instance file, to infer the believed truth value of this proposition, the believed truth values of those six XBRL items need to be inferred first.

First, as discussed in the previous section, each XBRL item is validated by checking whether this item is an item element in the specified XBRL instance file (lmt-20060930.xml); then, since this XBRL file is published in SEC's official website, the XBRL file is believed, so each XBRL item is also believed; by default assigned truth value, each item in the XBRL is claimed by the information creator to have assigned truth value of true; since an XBRL item is an asserted proposition, by the theorem KP-2, the believed truth value of each XBRL item is derived as true.

Given that the believed truth value of each support proposition is true, and this derived proposition is believed, by theorem kp-3, the believed truth value of the derived proposition "Argument-Posi-1" is true.

(3) Provenance reasoning on derived proposition "Argument-Posi-2", "Argument-Posi-3", "Argument-Posi-4", "Argument-Neg-1". By similar reasoning processes to "Argument-Posi-1", the believed truth values of these propositions can be derived as true.

(4) Provenance reasoning on derived proposition "RecommToBuy" (Continued). After the believed truth values of five support propositions are inferred as true, the believed truth value of this proposition can be derived as true.

7.4 Discussion

KP can be applied not only financial reports but also any other information service in finance such as news, financial digests, messages (electronic bulletin board) and blogs.

By using KP in financial information, investors can learn where the information comes from and determine whether this information can be believed. Reconsider the story of fraudulent corporation information in BBS, introduced in Chapter 1. If KP had been applied, people would have found that message comes from an untrusted information source, and that cheating event would not happen.

In this application case, the KP model used is static KP. Consider uncertainty in trust and truth values, we have applied uncertain KP model to this application case. Interested readers could refer to [90].

KP can help to make financial information more trustworthy and also help investors to judge the validity of these information.

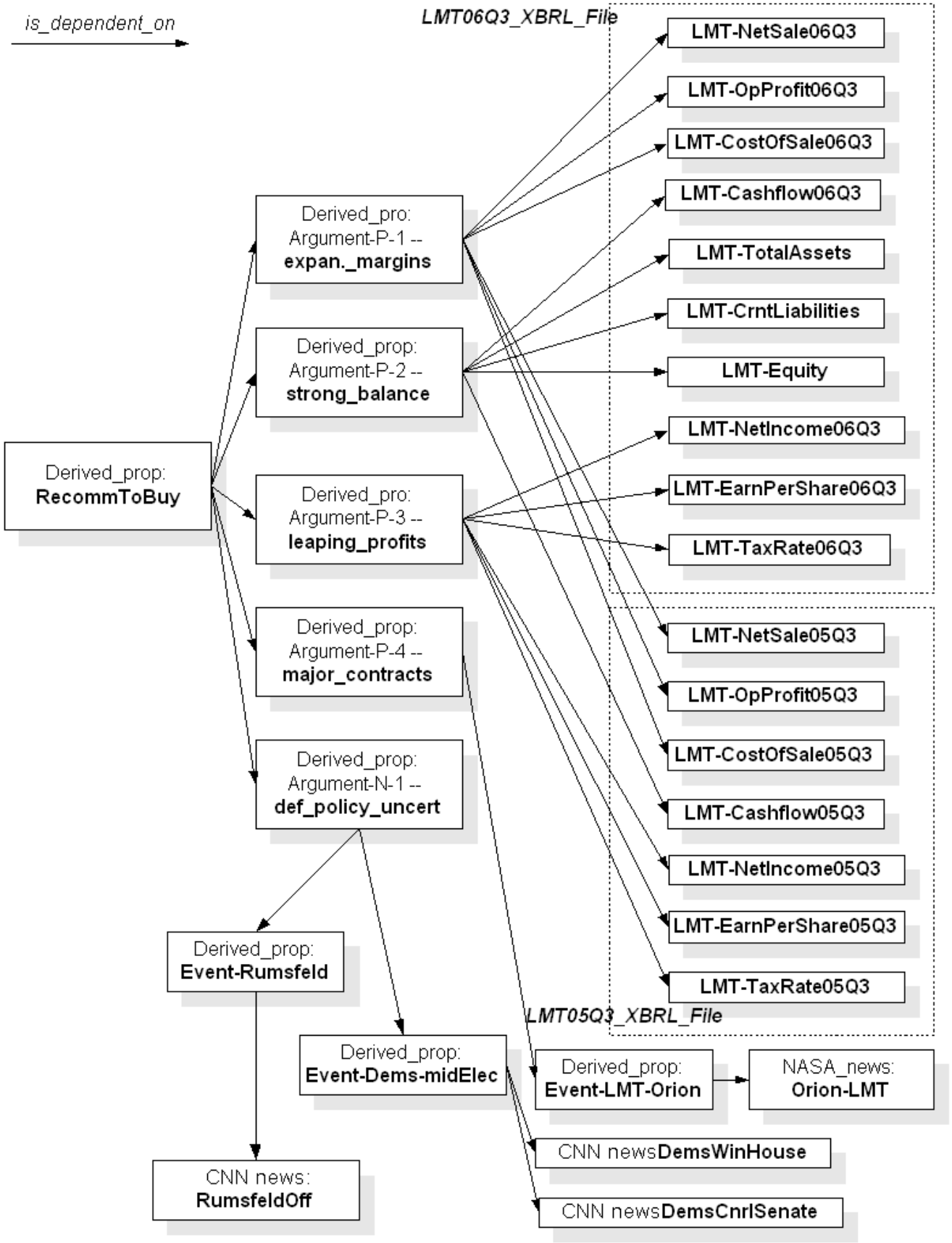

Figure 7.2: Dependency relations in the sample investment newsletter

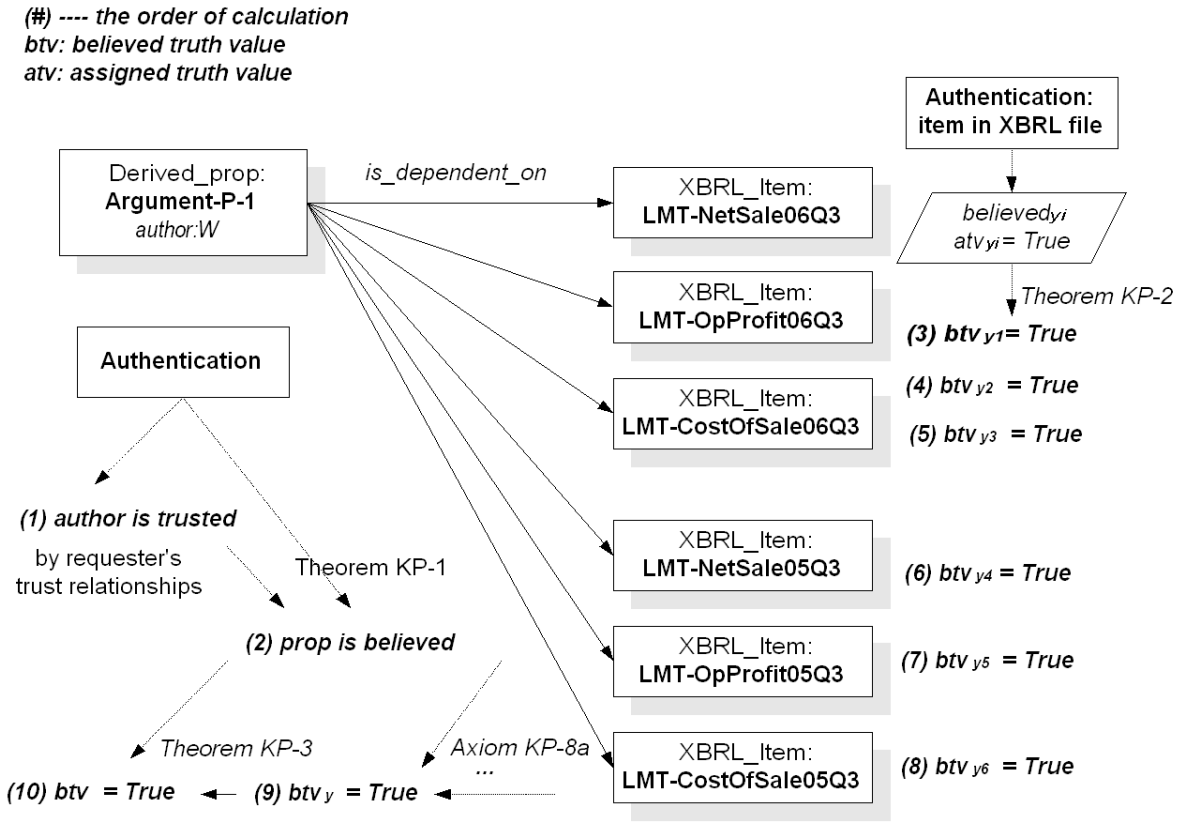

Figure 7.3: Provenance reasoning on proposition "Argument-posi-1"

Chapter 8

Summary and Future Work

Finally, in this chapter, we discuss the contribution, limitation, and future work.

8.1 Summary of Contributions

We have listed the major contributions of this thesis in Chapter 1. Here, we summarize the contribution again as follows. This thesis has two major contributions: (1) knowledge provenance; and (2) trust formalization.

8.1.1 Knowledge Provenance

The Web has become an open decentralized global information / knowledge repository. However, in this cyberspace, anyone can produce and disseminate information, so that the information may be true or false, current or outdated. How to discern the difference becomes a crucial problem on the Web.

This thesis first proposed knowledge provenance (KP) to address this problem, defined and axiomatized knowledge provenance ontology, to determine the origin and validity of information / knowledge on the Web, by means of modeling and maintaining information sources, information dependencies, and trust structures.

Knowledge provenance ontology is comprised of: (1) static KP, which formally specifies web information taxonomy, information sources, information dependencies, trust relationships, and provides a logical system to infer the origin and validity of web information. Static KP focuses on static and certain information, which lays a foundation for general knowledge provenance; (2)dynamic KP, which infers the validity of information that changes over time; (3) uncertainty-oriented KP, which infer the validity of information in a world with uncertain truth values and uncertain trust relationships. For length limitation, this thesis only covers static KP and dynamic KP. Uncertain KP can be found in [93] [87].

We also developed a web ontology of KP in OWL (the web ontology language for the Semantic Web), which can be used to annotate Web documents, then KP reasoners can crawl the web documents to collect provenance related attributes and infer the origin and validity of web information.

The use of KP will make Web information resources can be annotated with information sources and information dependencies, and will provide information users a tool to trace the origin and to determine the validity of Web information.

8.1.2 Formalizing Trust in Social Networks

Since the Web is widely used as a global information repository, a distributed computing platform, and global electronic markets, people and software agents need to interact with "strangers" on the Web. Can an entity trust another entity who is unknown before? Basically trust is established in the interaction between two entities. But, one entity only has a finite number of direct trust relationships, which cannot meet the needs of various interactions with unknown entities on the Web. As a promising remedy to this problem, social network based trust, in which A trusts B, B trusts C, thus A indirectly trusts C, is receiving considerable attention. A necessary condition for trust propagation in social networks is that trust need to be transitive. However, is trust transitive? What

types of trust are transitive and why? Few theories and models found so far answer these questions in a formal manner.

To fill the gap, this thesis constructed a logical theory of trust with situation calculus, in which the formal semantics of trust is defined; from the formal semantics, two types of trust – *trust in belief* and *trust in performance* were identified; the transitivity of *trust in belief* was revealed and proven; the conditions for trust propagation were derived. These results provide theoretical evidences to support trust propagation in social networks. In particular, this work reveals that *trust in belief* is transitive; *trust in performance* is not, but by *trust in belief*, *trust in performance* can propagate in social networks.

To facilitate trust reasoning using social networks in an easier straightforward form, based upon our proposed trust ontology, we also constructed a trust networks model, a graph representation for trust propagation in social networks, by which indirect trust reasoning in logic is transformed to simpler problem of trust path searching. The soundness and completeness of the trust networks model were proved.

Uncertainty widely exists in trust problems, especially in the cyberspace. To address this problem, we have extended our trust networks model to uncertain model, and revised theoretical condition for trust decision. They can be found in papers [88] [94]. For length limitation, this part of work is not included in this thesis.

Regarding social networks-based trust reasoning, most proposed models require to access to all personal trust data of the visited entities in social networks. However, for the reason of privacy, people usually do not publish their private data to the public. This thesis proposed a social network-based distributed trust reasoning model and constructed the model in situation calculus. This model can be implemented with web services, then each entity in social networks only answers a specific question of whether to trust a questioned entity, and the private trust data need not to be published online. In addition, this distributed model also makes trust path search in parallel.

On the issue of knowledge representation, our model has three advantages: (1) we

represent trust based on belief, a well studied concept in AI, which makes our model established on a concrete ground; (2) by using situation calculus as representation language, we are able to represent the context of trust as reified fluents. In this way, we found a solution to formally represent the context of trust; (3) the representation of trust in situation calculus also contributes to situation calculus for the language to describe trust among multiple agents. As we know, this is the first proposed model of trust in situation calculus.

8.2 Discussion

Same as other research, some limitations exist in this thesis, which are discussed as follows.

As stated in the beginning of Chapter 3, this thesis adopts the approach to determining the validity of information by provenance and trust. Therefore, KP does not analyze the content of information. For example, in the context of detecting email spam, instead of scanning the content of an email, KP makes judgment by considering who is the sender.

In KP, provenance reasoning is based on KP tags. KP models do not directly handle the validity of KP tags produced by information creators. In application, if an information creator intends to produce unreliable KP tags, the creator will get bad reputation, and information users will not trust this creator anymore.

Regarding trust, by the proposed formal trust model in this thesis, there is no limitation on the length of a trust path. However, in the real world, except the case of a chain of "absolute trust", trust will decay with the length of trust path. This problem can be solved in uncertain trust model, which is not covered by this thesis. In an application, a restriction may be applied to the length of trust path.

In addition, regarding the context of trust, on one hand, in order to prevent mistrust,

the context of a trust relationship should be very strict; on the other hand, if all the context of trust is specified very strictly, it may makes the length of a trust path very long, or makes the degree of the connectivity in a trust network largely decrease. Therefore, in general, people have to trade off the strictness and connectivity. For specific applications such as e-commerce, some standardized context for some typical cases could be a remedy.

8.3 Future Work

The research in this thesis can be extended in many directions. In future, we would like to continue our research in the following aspects.

Human-user Centered Knowledge Provenance

This thesis is aimed at developing a formal model of knowledge provenance for automatic judgment of the origin and validity of web information, which can be used by both human user and software agents. One of the directions for further development is human-user centered knowledge provenance, which could include human information users as decision analyzers. In this way, KP could consider a wider domain of provenance information such as the context of the original proposition. This type of information is difficult to be analyzed by machines, but it could be processed by human users, and this type of information could be important for people to judge the validity and the value of the questioned information.

Knowledge Provenance for Semantic Web

In our current models of KP, proposition is the most basic information unit. A KP proposition could be a piece of text or an XML element. Consider that the data in the semantic web, which are typically defined by RDF, RDFS and OWL, are also XML data. Therefore, the current models can be directly applied in the semantic web. On

the Semantic Web, some important information could be discovered by integrating the data spread over the Web; the trustworthiness of the discovery depends on the the provenance and the corresponding trustworthiness of each piece of information used for that derivation. Ding et al [42] depicts a picture to use knowledge provenance for information search, information integration and analysis on the Semantic Web.

From another view, with the semantic web, the knowledge representation model of RDF graph, which is a set of RDF triples of $< Object, Attribute, Value >$, makes knowledge provenance be able to look into to the construction of a KP proposition. This feature enables semantics analysis in KP. Some examples of such analysis are given as follows. It is possible to handle *Equivalent_prop* by semantics analysis; the requester may want to find whether the questioned proposition can be derived from a certain trusted knowledge sources; the requester may want to check whether the questioned proposition is consistent with the requester's knowledge base; the memberships of a proposition to knowledge fields can be determined automatically by analyzing the nodes in RDF graph. Semantics based approach of KP is much more sophisticated than current dependency and trust based approach.

How should revision type of dependency be handled?

In our current model of KP, we considered information dependencies in three types: derived propositions, equivalent propositions and compound propositions; however, how to handle revision type of dependency has not been addressed. Revision is a common and important type of dependency. For example, in wiki systems, a piece of information may evolves from many old versions. How to determine the validity of this type of information is still unclear. In future, we will explore the solutions for it.

System trust

As discussed in Chapter 2, system trust is the trust placed on the stable or predictable functions or behaviors of a system. System trust may appear as professional membership-based trust, characteristics-based trust, institution-based trust or regulation-based trust. This thesis does not cover system trust. In future, we would like to further develop the theory of trust to include system trust, particularly, characteristics-based trust. System trust is especially useful in the cases where a user has to interact with strangers and there are no trust pathes to them in social networks.

Trust evolution and personal trust management

Trust evolves in the interaction between two parties. Each individual in social networks adjusts the degree of trust in another individual according to whether the expectation in the trust is fulfilled. In future, we would like to model trust evolution and to develop personal trust management model to manage personal trust relationships and trust policies.

Trust management for distributed systems

As discussed in Chapter 5, the web has been becoming the platform for distributed computing, in which people and software agents need to interact with "strangers". For this reason, trust management is a crucial factor for distributed computing. Current trust management, such as KeyNote [16] or Web Services Language [173], only focuses on security, so it cannot meet the needs of trust judgment on the Web as we do in our real society. To fill the gap, we would like to apply the trust theory developed in this thesis into distributed systems. In particular, we are interested in developing a real e-business or e-services that integrates trust management in the semantic web services.

Healthcare Knowledge Provenance

Healthcare is a typical area where people are seriously concerned about the origin and validity of the information they get. Nowadays, there is a lot of healthcare related information on the web such as diet and nutrition, disease prevention, new treatments, and so forth. There are various information sources of different quality for different purposes. Some of them are professional and with high quality, but some others may be just for business purposes. The use of KP will help people to identify trustworthy information sources.

On the other hand, in the era of the information economy and globalization, enterprise integration is a strong trend for organizations to meet the era's demands of being able to promptly and seamlessly collaborate at low costs with a dynamic set of supply chain partners. Compared with manufacturing sector and many other business sectors, healthcare has fallen far behind on this issue. As the largest industrial sector in North America, there is a strong demand for healthcare to catch up, in order to improve the quality and efficiency of healthcare services and to handle the crises of pandemic diseases such as the avian flu. In enterprise integration, knowledge provenance can help to determine the validity of the shared information distributed along supply chains.

We will apply knowledge provenance in the healthcare field to develop healthcare knowledge provenance tools, and to develop best practices of healthcare knowledge provenance.

Appendix A

Knowledge Provenance Ontology in OWL

<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >

<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >

<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >

<!ENTITY owl "http://www.w3.org/2002/07/owl#" >

<!ENTITY foaf "http://xmlns.com/foaf/0.1/" >

<!ENTITY kp "http://eil.utoronto.ca/2006/06/kp#" >

]>

<rdf:RDF xmlns = "&kp;" xmlns:kp = "&kp;" xml:base = "http://eil.utoronto.ca/2006/06/kp"

xmlns:owl = "&owl;" xmlns:rdf = "&rdf;" xmlns:rdfs= "&rdfs;" xmlns:foaf= "&foaf;"

xmlns:xsd = "&xsd;" >

<Ontology rdf:about="">

<rdfs:comment>

Static Knowledge Provenance Ontology, Last Revised: 4 June 2007 </rdfs:comment>

<versionInfo> 2.0 </versionInfo>

</Ontology>

<owl:Class rdf:ID =“KPProp”>

<rdfs:label>

KPProp</rdfs:label>

<rdfs:comment> The following property is a derived property which is derived by a KP reasoner </rdfs:comment>

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“believedTruthValue”/>

<owl:cardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:cardinality>

<rdfs:comment> The truth value that the provenance requester believes this proposition having. Refers to KP axioms and theorems </rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“believedCertaintyDegree”/>

<owl:maxCardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:maxCardinality>

<rdfs:comment> Derived by uncertain KP reasoner. </rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“effectiveAt”/>

<owl:maxCardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:maxCardinality>

<rdfs:comment> A questioned time point; Derived by dynamic KP reasoner. </rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

</owl:Class>

<owl:Class rdf:ID =“OriginalProp”>

<rdfs:label>

OriginalProp</rdfs:label>

<rdfs:subClassOf rdf:resource =“&kp;KPProp”/>

<rdfs:comment> The following properties are defined-properties, which must be given in annotation of a web document. </rdfs:comment>

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“propContent”/>

<owl:cardinality rdf:datatype =“&xsd;nonNegativeInteger”> 0 </owl:cardinality>

<rdfs:comment> This property may be given in the form of the content of this OriginalProp</rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“hasAuthor”/>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“hasPublisher”/>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“inField”/>

<owl:minCardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:minCardinality>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“assignedTruthValue”/>

<owl:maxCardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:maxCardinality>

<rdfs:comment> This property is optional. If it is not given, the default value is

True </rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“assignedCertaintyDegree”/>

<owl:maxCardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:maxCardinality>

<rdfs:comment> This property is optional. If it is not given, the default value is

1.0. </rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“madeAt”/>

<owl:maxCardinality rdf:datatype = “&xsd;nonNegativeInteger”> 1 </owl:maxCardinality>

<rdfs:comment> The time of creation of the information </rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource = “effectiveFrom”/>

<owl:maxCardinality rdf:datatype = “&xsd;nonNegativeInteger”> 1 </owl:maxCardinality>

<rdfs:comment> This property is optional. If it is not given, the default value is

-inf. </rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“effectiveTo”/>

<owl:maxCardinality rdf:datatype = “&xsd;nonNegativeInteger”> 1 </owl:maxCardinality>

<rdfs:comment> This property is optional. If it is not given, the default value is

+inf. </rdfs:comment>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:comment> The following properties are derived properties, which are derived by a KP reasoner </rdfs:comment>

<rdfs:subClassOf >
<owl:Restriction>
<owl:onProperty rdf:resource =“hasAuthenticSource”/>
<owl:minCardinality rdf:datatype = “&xsd;nonNegativeInteger”> 0 </owl:minCardinality>
<rdfs:comment> An authentic source is an authentic author or publisher whose digital signature is validated successcefully</rdfs:comment>
</owl:Restriction>
</rdfs:subClassOf >

<rdfs:subClassOf >
<owl:Restriction>
<owl:onProperty rdf:resource =“believed”/>
<owl:cardinality rdf:datatype = “&xsd;nonNegativeInteger”> 1 </owl:cardinality>
<rdfs:comment> An original proposition is believed, if one of its authentic source is trusted by the provenance requester in a field which this proposition belongs to </rdfs:comment>
</owl:Restriction>
</rdfs:subClassOf >
</owl:Class>

<owl:Class rdf:ID =“DependentProp”>
<rdfs:label>
DependentProp</rdfs:label>
<rdfs:subClassOf rdf:resource = “&kp;KPProp”/>

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“isDependentOn”/>

<owl:minCardinality rdf:datatype = “&xsd;nonNegativeInteger”> 1 </owl:minCardinality>

</owl:Restriction>

</rdfs:subClassOf >

</owl:Class>

<owl:Class rdf:ID =“AssertedProp”>

<rdfs:label> AssertedProp </rdfs:label>

<rdfs:subClassOf rdf:resource = “&kp;OriginalProp”/>

</owl:Class>

<owl:Class rdf:ID =“DerivedProp”>

<rdfs:label>

DerivedProp</rdfs:label>

<rdfs:subClassOf rdf:resource = “&kp;OriginalProp”/>

<rdfs:subClassOf rdf:resource = “&kp;DependentProp”/>

</owl:Class>

<owl:Class rdf:ID =“EquivalentProp”>

<rdfs:label>

EquivalentProp</rdfs:label>

<rdfs:subClassOf rdf:resource = “&kp;DependentProp”/>

<rdfs:comment> The following properties are defined-properties, which must be given in annotation of a web document. </rdfs:comment>

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“propContent”/>

<owl:cardinality rdf:datatype = “&xsd;nonNegativeInteger”> 1 </owl:cardinality>

</owl:Restriction>

</rdfs:subClassOf >

</owl:Class>

<owl:Class rdf:ID =“CompoundProp”>

<rdfs:label> CompoundProp </rdfs:label>

<rdfs:subClassOf rdf:resource = “&kp;DependentProp”/>

</owl:Class>

<owl:Class rdf:ID =“AndProp”>

<rdfs:label> AndProp </rdfs:label>

<rdfs:subClassOf rdf:resource = “&kp;CompoundProp”/>

</owl:Class>

<owl:Class rdf:ID =“OrProp”>

<rdfs:label> OrProp </rdfs:label>

<rdfs:subClassOf rdf:resource = “&kp;CompoundProp”/>

</owl:Class>

<owl:Class rdf:ID =“NegProp”>

<rdfs:label> NegProp </rdfs:label>

<rdfs:subClassOf rdf:resource = “&kp;CompoundProp”/>

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“isDependentOn”/>

<owl:cardinality rdf:datatype = “&xsd;nonNegativeInteger”> 1 </owl:cardinality>

</owl:Restriction>

</rdfs:subClassOf >

</owl:Class>

<owl:ObjectProperty rdf:ID =“propContent”>

<rdfs:label> propContent</rdfs:label>

<rdfs:domain>

<owl:Class>

<owl:unionOf rdf:parseType=“Collection”>

<owl:Class rdf:about=“&kp;OriginalProp”/>

<owl:Class rdf:about=“&kp;EquivalentProp”/>

</owl:unionOf>

</owl:Class>

</rdfs:domain>

</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID =“inField”>

<rdfs:label> inField </rdfs:label>

<rdfs:domain rdf:resource =“&kp;OriginalProp” />

</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID =“hasAuthor”>

<rdfs:label> hasAuthor </rdfs:label>

<rdfs:subPropertyOf rdf:resource =“hasInfoCreator”/>

<rdfs:domain rdf:resource =“&kp;OriginalProp” />

<rdfs:range rdf:resource =“&foaf;Agent” />

</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID =“hasPublisher”>

<rdfs:label> hasPublisher </rdfs:label>

<rdfs:subPropertyOf rdf:resource =“hasInfoCreator”/>

<rdfs:domain rdf:resource =“&kp;OriginalProp” />

<rdfs:range rdf:resource =“&foaf;Agent” />

</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID =“hasAuthenticSource”>

<rdfs:label> hasAuthenticSource </rdfs:label>

<rdfs:domain rdf:resource =“&kp;OriginalProp” />

<rdfs:range rdf:resource =“&foaf;Agent” />

</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID =“isDependentOn”>

<rdfs:label> isDependentOn</rdfs:label>

<rdfs:domain rdf:resource =“&kp;DependentProp” />

<rdfs:range rdf:resource =“&kp;KPProp” />

</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID =“assignedTruthValue”>

<rdfs:label> assignedTruthValue </rdfs:label>

<rdfs:domain rdf:resource =“&kp;OriginalProp” />

<rdfs:range rdf:resource =“&xsd;boolean” />

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID =“believed”>

<rdfs:label> believed </rdfs:label>

<rdfs:domain rdf:resource =“&kp;OriginalProp” />

<rdfs:range rdf:resource =“&xsd;boolean” />

</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID ="believedTruthValue">

<rdfs:label> Trusted Truth Value </rdfs:label>

<rdfs:domain rdf:resource ="&kp;KPProp" />

</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID ="assignedCertaintyDegree">

<rdfs:label> assignedCertaintyDegree </rdfs:label>

<rdfs:comment>

Degree of belief, assigned by information creator; used for uncertain KP </rdfs:comment>

<rdfs:domain rdf:resource ="&kp;OriginalProp" />

<rdfs:range rdf:resource ="&xsd;float" />

<rdfs:comment>

The range changes from 0 to 1.0</rdfs:comment>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID ="believedCertaintyDegree">

<rdfs:label> believedCertaintyDegree </rdfs:label>

<rdfs:comment>

Degree of belief, derived by KP reasoner representing the information user; used for uncertain KP </rdfs:comment>

<rdfs:domain rdf:resource ="&kp;KPProp" />

<rdfs:range rdf:resource ="&xsd;float" />

<rdfs:comment>

The range changes from 0 to 1.0</rdfs:comment>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID ="madeAt">

<rdfs:label> madeAt </rdfs:label>

<rdfs:domain rdf:resource ="&kp;OriginalProp" />

<rdfs:range rdf:resource =“&xsd;dateTime” />

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID =“effectiveFrom”>

<rdfs:label> effectiveFrom </rdfs:label>

<rdfs:domain rdf:resource =“&kp;OriginalProp” />

<rdfs:range rdf:resource =“&xsd;dateTime” />

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID =“effectiveTo”>

<rdfs:label> effectiveTo </rdfs:label>

<rdfs:domain rdf:resource =“&kp;OriginalProp” />

<rdfs:range rdf:resource =“&xsd;dateTime” />

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID =“effectiveAt”>

<rdfs:label> effectiveAt </rdfs:label>

<rdfs:domain rdf:resource =“&kp;KPProp” />

<rdfs:range rdf:resource =“&xsd;dateTime” />

</owl:DatatypeProperty>

</rdf:RDF>

Appendix B

Trust Ontology in OWL

<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >

<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >

<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >

<!ENTITY owl "http://www.w3.org/2002/07/owl#" >

<!ENTITY tr "http://eil.utoronto.ca/kp/2006/06/tr#" >

<!ENTITY foaf "http://xmlns.com/foaf/0.1/" >

]>

<rdf:RDF xmlns ="&tr;" xmlns:tr ="&tr;" xml:base ="&tr;" xmlns:rdf ="&rdf;" xmlns:rdfs ="&rdfs;" xmlns:owl ="&owl;" xmlns:xsd ="&xsd;" >

<Ontology rdf:about="">

<rdfs:comment>

Trust ontology</rdfs:comment>

<versionInfo>

06 July 2006 Revised</versionInfo>

```
</Ontology>

<owl:Class rdf:about ="Trustor">
<rdfs:subClassOf rdf:resource ="&foaf;Agent"/>
<rdfs:subClassOf >
<owl:Restriction>
<owl:onProperty rdf:resource ="hasTrustRelationship"/>
<owl:minCardinality rdf:datatype ="&xsd;nonNegativeInteger"> 1 </owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf >
</owl:Class>

<owl:ObjectProperty rdf:about ="hasTrustRelationship">
<rdfs:label> Has Trust Relationship </rdfs:label>
<rdfs:domain rdf:resource ="&tr;Trustor" />
<rdfs:range rdf:resource ="TrustRelationship" />
</owl:ObjectProperty>

<owl:Class rdf:ID="TrustRelationship">
<owl:unionOf rdf:parseType ="collection">
<owl:Class rdf:about ="TrustInBelief" />
<owl:Class rdf:about ="TrustInPerformance" />
</owl:unionOf >
</owl:Class>

<owl:Class rdf:rdf:ID="TrustInBelief">
<rdfs:subClassOf rdf:resource ="TrustRelationship"/>
<rdfs:subClassOf >
<owl:Restriction>
```

<owl:onProperty rdf:resource =“trustee”/>

<owl:allValueFrom rdf:resource =“&foaf;Agent” />

<owl:cardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:cardinality>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“inContext”/>

<owl:minCardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:minCardinality>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

<owl:Class rdf:rdf:ID=“TrustInPerformance”>

<rdfs:subClassOf rdf:resource =“TrustRelationship”/>

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“trustee”/>

<owl:allValueFrom rdf:resource =“&foaf;Agent” />

<owl:cardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:cardinality>

</owl:Restriction>

</rdfs:subClassOf >

<rdfs:subClassOf >

<owl:Restriction>

<owl:onProperty rdf:resource =“inContext”/>

<owl:minCardinality rdf:datatype =“&xsd;nonNegativeInteger”> 1 </owl:minCardinality>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

<owl:ObjectProperty rdf:ID =“trustee”>

<rdfs:domain rdf:resource =“TrustRelationship” />

<rdfs:range rdf:resource =“&foaf;Agent” />

</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID =“inContext”>

<rdfs:domain rdf:resource =“TrustRelationship” />

</owl:ObjectProperty>

</rdf:RDF>

Appendix C

Proof of Theorems

C.1 Chapter 3 Static KP

Theorem KP-1:

$$type(x, Original\_prop) \\ \wedge has\_authentic\_source(x, c, a) \wedge trusted\_in(a, c, f) \\ \wedge in\_field(x, f)) \\ \supset believed(x, a). \quad \text{(C.1)}$$

Proof: In the following, all variables are universally quantified in the largest scope. Axiom KP-3 (3.22) is logically equivalent to

$$\neg trusted\_in(a, c, f) \vee \\ (\neg(type(x, Original\_prop) \wedge has\_authentic\_source(x, c, a) \wedge in\_field(x, f)) \\ \vee believed(x, a)), \quad \text{(C.2)}$$

which is logically equivalent to

$$\neg(trusted\_in(a, c, f) \wedge type(x, Original\_prop) \wedge has\_authentic\_source(x, c, a) \wedge in\_field(x, f)) \vee believed(x, a)). \quad \text{(C.3)}$$

This is logically equivalent to Theorem KP-1.

□

Theorem KP-2:

$$type(x, Asserted\_prop) \wedge believed(x, a) \wedge assigned\_truth\_value(x, v) \supset believed\_truth\_value(a, x, v). \quad \text{(C.4)}$$

Proof: Axiom KP-4 is logically equivalent to

$$\neg(type(x, Asserted\_prop) \wedge believed(x, a)) \vee (\neg assigned\_truth\_value(x, v) \vee believed\_truth\_value(a, x, v)), \quad \text{(C.5)}$$

which is further logically equivalent to

$$\neg(type(x, Asserted\_prop) \wedge believed(x, a) \wedge assigned\_truth\_value(x, v)) \vee believed\_truth\_value(a, x, v)). \quad \text{(C.6)}$$

This is logically equivalent to Theorem KP-2. All variables are universally quantified in the largest scope.

□

Theorem KP-3:

$$type(x, Derived\_prop) \wedge assigned\_truth\_value(x, v) \wedge believed(x, a) \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, True) \supset believed\_truth\_value(a, x, v). \quad (C.7)$$

Proof: In the following, all variables are universally quantified in the largest scope. Axiom KP-5 is logically equivalent to

$$\neg(type(x, Derived\_prop) \wedge is\_dependent\_on(x, y) \wedge believed(x, a)) \vee ((\neg believed\_truth\_value(a, y, True) \vee (\neg assigned\_truth\_value(x, v) \vee believed\_truth\_value(a, x, v))); \quad (C.8)$$

this is logically equivalent to

$$\neg(type(x, Derived\_prop) \wedge is\_dependent\_on(x, y) \wedge believed(x, a)) \vee (\neg(believed\_truth\_value(a, y, True) \wedge assigned\_truth\_value(x, v)) \vee believed\_truth\_value(a, x, v)); \quad (C.9)$$

further logically equivalent to

$$\neg(type(x, Derived\_prop) \wedge is\_dependent\_on(x, y) \wedge believed(x, a) \wedge believed\_truth\_value(a, y, True) \wedge assigned\_truth\_value(x, v)) \vee believed\_truth\_value(a, x, v). \quad (C.10)$$

This is logically equivalent to the theorem.

□

Theorem KP-4: *By system $\mathcal{T}_{KP1}$, given $\mathcal{KB}_{KP1,facts}$ that satisfies $\mathcal{T}_{KP1,KB}$, for any provenance requester a, any KP_prop x has one and only one believed truth value of either*

"True", "False" or "Unknown".

$$\mathcal{KB}_{KP1,rules} \models$$

$$\mathcal{KB}_{KP1,facts} \supset$$

$$(type(x, KP\_prop) \supset$$

$$((believed\_truth\_value(a, x, True) \vee believed\_truth\_value(a, x, False)$$

$$\vee believed\_truth\_value(a, x, Unknown))$$

$$\wedge \neg(believed\_truth\_value(x, True) \wedge believed\_truth\_value(x, False))$$

$$\wedge \neg(believed\_truth\_value(x, True) \wedge believed\_truth\_value(x, Unknown))$$

$$\wedge \neg(believed\_truth\_value(x, False) \wedge believed\_truth\_value(x, Unknown))))$$

(C.11)

Proof: We need to prove one and only one of the following three formulas to be true.

$$believed\_truth\_value(a, x, True)$$

$$believed\_truth\_value(a, x, False)$$

$$believed\_truth\_value(a, x, Unknown)$$

By Axiom KP-11, a KP_prop instance is of exactly one of the six basic types: Asserted_prop, Derived_prop, Equivalent_prop, Neg_prop, And_prop and Or_prop. Therefore, these six basic types form a partition of all possible cases. We can prove the theorem in these six cases one by one.

By the KP_prop taxonomy, among six basic types, except Asserted_prop, all others are Dependent_prop. In $\mathcal{T}_{KP1}$, the conditions to define the believed truth value of a Dependent_prop contain the believed truth value of the proposition that that Dependent_prop depends on. Because of this recursion, we need apply strong mathematical

induction to the dependency length[1], to prove the theorem.

In the following, we first prove the theorem in the case of Asserted_props, and then prove the cases of Dependent_props.

(1) When x is an Asserted_prop, theorem KP-2 and axiom KP-10 are the only two rules used to infer x's believed truth value. By theorem KP-2, the condition to determine a deterministic believed truth value for x is

$$believed(x, a) \wedge assigned\_truth\_value(x, v).$$

By axiom KP-12,

$$assigned\_truth\_value(x, v)$$

must be true, and v is bound to either "True" or "False".

So, if

$$believed(x, a)$$

is true, $v = True$ will make

$$believed\_truth\_value(a, x, True)$$

true. By KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, False)$$

and

$$\neg believed\_truth\_value(a, x, Unknown)$$

are derived; and $v = False$ will make

$$believed\_truth\_value(a, x, False)$$

[1]Dependency length refers to the number of dependency levels to link to Asserted_prop(s). For example, A is an Asserted_prop, if B is dependent on A, then the dependency length of B is 1; if C is further dependent on B, then the dependency length of C is 2, and so forth.

true, and both

$$\neg believed\_truth\_value(a, x, True)$$

and

$$\neg believed\_truth\_value(a, x, Unknown)$$

are derived; otherwise, i.e.

$$believed(x, a)$$

is false, by KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, True)$$

and

$$\neg believed\_truth\_value(a, x, False)$$

are derived. By axiom KP-10,

$$believed\_truth\_value(a, x, Unknown)$$

is derived.

Therefore, when x is an Asserted_prop, it has one and only one believed truth value of "True", "False", or "Unknown".

(2) When x is a Dependent_prop, we apply strong mathematical induction to the dependency length.

(2.1) Consider the base case, the dependency length is 1, that is, a Dependent_prop is directly dependent on Asserted_prop(s). Now, we consider the 5 cases of basic types of Dependent_props.

(2.1.1) when x is a Derived_prop, theorem KP-3 and axiom KP-10 are all the rules to infer the believed truth value of x. By theorem KP-3, the condition to determine a deterministic believed truth value for x is

$$believed(x, a) \wedge is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, True) \wedge assigned\_truth\_value(x, v)$$

By axiom KP-14, x must have one and only one support proposition y, i.e.

$$is\_dependent\_on(x, y)$$

must be true and this support proposition y is unique.

By the base assumption (given in 2.1) that y is an Asserted_prop, and the conclusion of case (1), we know y must have one and only one believed truth value of either "True", "False", or "Unknown".

By axiom KP-12,

$$assigned\_truth\_value(x, v)$$

must be true, and v is bound to either "True" or "False".

So, if

$$believed(x, a) \wedge believed\_truth\_value(a, y, True)$$

is true, $v = True$ will make

$$believed\_truth\_value(a, x, True)$$

true, and by KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, False)$$

and

$$\neg believed\_truth\_value(a, x, Unknown)$$

are derived; and $v = False$ will make

$$believed\_truth\_value(a, x, False)$$

true, and both

$$\neg believed\_truth\_value(a, x, True)$$

and

$$\neg believed\_truth\_value(a, x, Unknown)$$

are derived; otherwise, by KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, True)$$

and

$$\neg believed\_truth\_value(a, x, False)$$

are derived. By axiom KP-10,

$$believed\_truth\_value(a, x, Unknown)$$

is derived.

Thus, in the base case, when x is a Derived_prop, the theorem is true.

(2.1.2) when x is an Equivalent_prop, axiom KP-6 and axiom KP-10 are all the rules to infer the believed truth value of x. By axiom KP-6, the condition to determine the believed truth value of x is

$$is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, v)$$
$$\wedge prop\_content(x, c_1) \wedge prop\_content(y, c_2) \wedge equivalent\_to(c_1, c_2))$$

Similar to (2.1.1), by axiom KP-14, x must have one and only one support proposition y, i.e.

$$is\_dependent\_on(x, y)$$

must be true and this support proposition y is unique.

By the base assumption (given in 2.1) that y is an Asserted_prop, and the conclusion of case (1), we know

$$believed\_truth\_value(a, y, v)$$

must be true, and v is bound to either "True", "False", or "Unknown".

By axiom KP-13, x must have unique content c_1, and y must have a unique content c_2, that is,

$$prop\_content(x, c_1) \wedge prop\_content(y, c_2)$$

must be true, and c_1 and c_2 are bound to a proposition content separately.

So, if

$$equivalent\_to(c_1, c_2)$$

is true, then

$$believed\_truth\_value(a, x, v)$$

is derived, v is the same value as y's believed truth value. This means

$$believed\_truth\_value(a, x, v)$$

is true, and by KCA and NF-rule, for all $v' \neq v$ & $v' \in$ {True, False, Unknown}

$$\neg believed\_truth\_value(a, x, v')$$

is derived; otherwise, by KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, True)$$

and

$$\neg believed\_truth\_value(a, x, False)$$

are derived. By axiom KP-10,

$$believed\_truth\_value(a, x, Unknown)$$

is derived.

Thus, in the base case, when x is an Equivalent_prop, the theorem is true.

(2.1.3) when x is an Neg_prop, axiom KP-7 and axiom KP-10 are all the rules to infer the believed truth value of x. By axiom KP-7, the condition to determine the believed truth value of x is

$$is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, v)$$

Similar to (2.1.2), by axiom KP-14, x must have one and only one support proposition y, i.e.

$$is\_dependent\_on(x, y)$$

must be true and this support proposition y is unique; by the base assumption (given in 2.1) that y is an Asserted_prop, and the conclusion of case (1), we know

$$believed\_truth\_value(a, y, v)$$

must be true, and v is bound to either "True", "False", or "Unknown".

If $v \in$\{True,False\},

$$believed\_truth\_value(a, x, neg(v))$$

is derived to be true, and by KCA and NF-rule, for all $v' \neq neg(v)$ & $v' \in$ {True, False, Unknown}

$$\neg believed\_truth\_value(a, x, v')$$

is derived; otherwise, by KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, True)$$

and

$$\neg believed\_truth\_value(a, x, False)$$

are derived. By axiom KP-10,

$$believed\_truth\_value(a, x, Unknown)$$

is derived.

Thus, in the base case, when x is an Neg_prop, the theorem is true.

(2.1.4) when x is an And_prop, axiom KP-8 and axiom KP-10 are all the rules to infer the believed truth value of x. By axiom KP-8a, the condition to derive the believed

truth value of "True" is

$$is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge y_1 \neq y_2$$
$$\wedge\, believed\_truth\_value(a, y_1, True) \wedge believed\_truth\_value(a, y_2, True);$$

and the condition to derive the believed truth value of "False" is

$$is\_dependent\_on(x, y) \wedge believed\_truth\_value(a, y, False)$$

By axiom KP-14, x must have exactly two different support propositions y_1 and y_2, i.e.

$$is\_dependent\_on(x, y_1) \wedge is\_dependent\_on(x, y_2) \wedge y_1 \neq y_2$$

must be true.

By the base assumption (given in 2.1) that y is an Asserted_prop, and the conclusion of case (1), we know y_1 and y_2 must have one and only one believed truth value of either "True", "False", or "Unknown".

If

$$believed\_truth\_value(a, y_1, True) \wedge believed\_truth\_value(a, y_2, True)$$

is true, then

$$believed\_truth\_value(a, x, True)$$

is derived to be true, and by KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, False)$$

and

$$\neg believed\_truth\_value(a, x, Unknown)$$

are derived; otherwise if one of y_1 and y_2, say y_1, has believed truth value of "False", i.e.

$$believed\_truth\_value(a, y_1, False)$$

is true, then

$$believed\_truth\_value(a, x, False)$$

is derived to be true, and by KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, True)$$

and

$$\neg believed\_truth\_value(a, x, Unknown)$$

are derived; otherwise, by KCA and NF-rule, both

$$\neg believed\_truth\_value(a, x, True)$$

and

$$\neg believed\_truth\_value(a, x, False)$$

are derived. By axiom KP-10,

$$believed\_truth\_value(a, x, Unknown)$$

is derived.

Thus, in the base case, when x is an And_prop, the theorem is true.

(2.1.5) when x is an Or_prop, the theorem can also be proved to be true in the base case. The proof is very similar to the case of And_prop and is omitted.

(2.2) Assume that for dependency length $\leq n$, a Dependent_prop has one and only one believed truth value of either "True", "False" or "Unknown".

(2.3) Now, we prove for dependency length $= n + 1$, a Dependent_prop also has one and only one believed truth value of either "True", "False" or "Unknown".

By axiom KP-14, there is no dependency loop. Since this Dependent_prop has dependency length of $n+1$, the dependency length of all its support proposition(s) must be less than or equivalent to n. By assumption in (2.2), all its support proposition(s) must have one and only one believed truth value of either "True", "False" or "Unknown".

Similar to the proof in the base case (given in 2.1), when all its support proposition(s) have one and only one believed truth value of either "True", "False" or "Unknown", this Dependent_prop also does.

By the strong mathematical induction, this theorem is proved.

□

C.2 Chapter 5 Trust

Some of theorems in this chapter are proved in a formal form. The notation used for a formal proof is as follows.

Each step of proof comprises 5 parts: (1) line number; (2) proof; (3) reasons; (4) premises used; (5) free variables in premises used; P denotes rule for premises.

T denotes of rule for tautologies; CP denotes rule of conditional proof; US denotes rule of universal specification; UG denotes rule of universal generalization; and so forth.

Proposition TR-1: Given any fluents p and q,

$$entail(p\dot{\wedge}q, p). \tag{C.12}$$

Proof: by the definition of predicate entail, the proposition to be proved is logically equivalent to

$$(\forall s)(holds(p\dot{\wedge}q, s) \supset holds(p, s)); \tag{C.13}$$

by the definitions of "propositional functions" of fluents (refer to (5.1)), the above formula is logically equivalent to

$$(\forall s)(holds(p, s) \wedge holds(q, s) \supset holds(p, s)); \tag{C.14}$$

This formula is a tautology. Therefore, this proposition is valid.

□

Proposition TR-2: Given any fluents p, q, and k,

$$entail(k, p) \wedge entail(p, q) \supset entail(k, q). \tag{C.15}$$

Proof: By the definition of *entail*, we need to prove

$$(\forall s)(holds(k, s) \supset holds(q, s)),$$

from

$$entail(k, p) \wedge entail(p, q).$$

We prove this proposition by indirect proof. Assume

$$\neg(\forall s)(holds(k, s) \supset holds(q, s)),$$

that is,

$$(\exists s)\neg(holds(k, s) \supset holds(q, s)).$$

Let situation z is such a situation, at which

$$\neg(holds(k, z) \supset holds(q, z)),$$

that is,

$$holds(k, z) \wedge \neg holds(q, z)).$$

Since $holds(k, z)$, by $entail(k, p)$,, we have

$$holds(p, z);$$

furthermore, by $entail(p, q)$, we have

$$holds(q, z).$$

This is a contradiction to the earlier assumption. Therefore, we proved

$$(\forall s)(holds(k, s) \supset holds(q, s)),$$

i.e.

$$entail(k, q),$$

from

$$entail(k, p) \wedge entail(p, q).$$

□

Proposition TR-4

$$holds(believe(d, k\dot{\supset}x), s) \wedge entail(q, k) \supset holds(believe(d, q\dot{\supset}x), s) \tag{C.16}$$

Proof: we prove this proposition by deriving

$$holds(believe(d, q\dot{\supset}x), s)$$

from

$$holds(believe(d, k\dot{\supset}x), s) \wedge entail(q, k).$$

First, by definition TR-2,

$$entail(q, k)$$

is logically equivalent to

$$(\forall z)(holds(q, z) \supset holds(k, z));$$

by definition TR-1, this is logically equivalent to

$$\forall z, holds(q\dot{\supset}k, z);$$

by *necessitation rule*, we have

$$holds(believe(d, q\dot{\supset}k), s);$$

now, from this formula and another premise

$$holds(believe(d, k\dot{\supset}x), s),$$

by axiom TR-2, we have

$$holds(believe(d, (q \dot{\supset} k) \wedge (k \dot{\supset} x)), s),$$

by the definition of "propositional functions" of fluents (definition TR-1), we obtain

$$holds(believe(d, k \dot{\supset} x), s).$$

□

Theorem TR-1.

$$(\forall x)(holds(trust\_p(d, e, x, k), s)) \wedge holds(made(y, e, q), s) \wedge entail(q, k) \supset holds(believe(d, k \dot{\supset} y), s). \quad \text{(C.17)}$$

Proof: We need to prove:

$$TR \vDash (\forall d, e, k, s, y, q)((\forall x)(holds(trust\_p(d, e, x, k), s)) \wedge holds(made(y, e, q), s) \wedge entail(q, k) \supset holds(believe(d, k \dot{\supset} y), s)),$$

where TR denotes the set of axioms and propositions defined for Trust ontology. We give a formal proof as follows.

(1)

$$(\forall x)(holds(trust\_p(d, e, x, k), s)) \wedge holds(made(y, e, q), s) \wedge entail(q, k);$$

by rule P (for CP); used premises: {(1)}; free variables in used premises: {*d,e,k,s,y,q*}.

(2)

$$(\forall x)(holds(trust\_p(d, e, x, k), s));$$

by (1) and rule T; used premises: $\{(1)\}$; free variables in used premises: $\{d,e,k,s\}$.

(3)

$$holds(made(y, e, q), s);$$

by (1) and rule T; used premises: $\{(1)\}$; free variables in the used premises: $\{y,e,q,s\}$.

(4)

$$entail(q, k);$$

by (1) and rule T; used premises: $\{(1)\}$; free variables in the used premises: $\{q,k\}$.

(5)

$$(\forall d', e', x', k', s')(holds(trust\_p(d', e', x', k'), s') \equiv$$
$$\forall q', (holds(made(x', e', q'), s') \wedge entail(q', k') \supset holds(believe(d', k' \dot{\supset} x'), s')));$$

by rule P (introducing Axiom TR-3); used premise: $\{(5)\}$; no free variable.

(6)

$$(\forall x')(holds(trust\_p(d, e, x', k), s) \equiv$$
$$(\forall q')(holds(made(x', e, q'), s) \wedge entail(q', k) \supset holds(believe(d, k \dot{\supset} x'), s)));$$

by (5) and rule US; used premise: $\{(5)\}$.

(7)

$$(\forall x')(holds(trust\_p(d, e, x', k), s)) \equiv$$
$$(\forall x')(\forall q')(holds(made(x', e, q'), s) \wedge entail(q', k) \supset holds(believe(d, k \dot{\supset} x'), s)));$$

by (6) and rule T; used premise: $\{(5)\}$.

(8)

$$(\forall x')(\forall q')(holds(made(x', e, q'), s) \wedge entail(q', k) \supset holds(believe(d, k\dot{\supset}x'), s));$$

by (2),(7) and rule T; used premise: $\{(1),(5)\}$; free variables in used premises: $\{d,e,k,s\}$.

(9)

$$holds(made(y, e, q), s) \wedge entail(q, k) \supset holds(believe(d, k\dot{\supset}y), s);$$

by (8) and rule *US*, {y/x}; used premise: $\{(1),(5)\}$; free variables in used premises: $\{d,e,k,s,y,q\}$.

(10)

$$holds(believe(d, k\dot{\supset}y), s);$$

by (9), (3), (4) and rule T; used premise: $\{(1),(5)\}$; free variables in used premises: $\{d,k,y,s\}$.

(11)

$$(\forall x)(holds(trust\_p(d, e, x, k), s)) \wedge holds(made(y, e, q), s) \wedge entail(q, k) \supset holds(believe(d, k\theta\dot{\supset}y), s)$$

by (1), (11) and rule *CP*; used premise: $\{(5)\}$.

(12)

$$(\forall d, e, k, s, y, q)((\forall x)(holds(trust\_p(d, e, x, k), s)) \wedge holds(made(y, e, q), s) \wedge entail(q, k) \supset holds(believe(d, k\theta\dot{\supset}y), s)),$$

by (11) and rule UG; used premise: $\{(5)\}$.

Since (5) Axiom TR-3 is one of the axiom defined in Trust ontology, this theorem is proved.

Theorem TR-2.

$$(\forall x)(holds(trust\_b(d,e,x,k),s)) \wedge holds(believe(e,q\dot{\supset}y),s) \wedge entail(q,k) \supset holds(believe(d,k\dot{\supset}y),s) \quad \text{(C.18)}$$

Proof: We need to prove:

$$TR \vDash (\forall d,e,k,s,y,q)((\forall x)(holds(trust\_b(d,e,x,k),s)) \wedge holds(believe(e,q\dot{\supset}y),s) \wedge entail(q,k) \supset holds(believe(d,k\dot{\supset}y),s)),$$

where TR denotes the set of axioms and propositions defined for Trust ontology. We give a formal proof as follows.

(1)

$$(\forall x)(holds(trust\_b(d,e,x,k),s)) \wedge holds(believe(e,q\dot{\supset}y),s) \wedge entail(q,k);$$

by rule P (for CP); used premises: $\{(1)\}$; free variables in used premises: $\{d,e,k,s,y,q\}$.

(2)

$$(\forall x)(holds(trust\_b(d,e,x,k),s));$$

by (1) and rule T; used premises: $\{(1)\}$; free variables in used premises: $\{d,e,k,s\}$.

(3)

$$holds(believe(e, q\dot{\supset}y), s);$$

by (1) and rule T; used premises: {(1)}; free variables in the used premises: {y,e,q,s}.

(4)

$$entail(q, k);$$

by (1) and rule T; used premises: {(1)}; free variables in the used premises: {q,k}.

(5)

$$(\forall d', e', x', k', s')(holds(trust\_b(d', e', x', k'), s') \equiv$$
$$\forall q', (holds(believe(e', q'\dot{\supset}x'), s') \wedge entail(q', k') \supset holds(believe(d', k'\dot{\supset}x'), s')));$$

by rule P (introducing Axiom TR-4); used premise: {(5)}; no free variable.

(6)

$$(\forall x')(holds(trust\_b(d, e, x', k), s) \equiv$$
$$(\forall q')(holds(believe(e, q'\dot{\supset}x'), s) \wedge entail(q', k) \supset holds(believe(d, k\dot{\supset}x'), s)));$$

by (5) and rule US; used premise: {(5)}.

(7)

$$(\forall x')(holds(trust\_b(d, e, x', k), s)) \equiv$$
$$(\forall x')(\forall q')(holds(believe(e, q'\dot{\supset}x'), s) \wedge entail(q', k) \supset holds(believe(d, k\dot{\supset}x'), s)));$$

by (6) and rule T; used premise: {(5)}.

(8)

$$(\forall x')(\forall q')(holds(believe(e, q'\dot{\supset}x'), s) \wedge entail(q', k) \supset holds(believe(d, k\dot{\supset}x'), s));$$

by (2),(7) and rule T; used premise: {(1),(5)}; free variables in used premises: {*d,e,k,s*}.

(9)

$$holds(believe(e, q\dot{\supset}y), s) \wedge entail(q, k) \supset holds(believe(d, k\dot{\supset}y), s);$$

by (8) and rule *US*, {y/x}; used premise: {(1),(5)}; free variables in used premises: {*d,e,k,s,y,q*}.

(10)

$$holds(believe(d, k\dot{\supset}y), s);$$

by (9), (3), (4) and rule T; used premise: {(1),(5)}; free variables in used premises: {*d,k,y,s*}.

(11)

$$(\forall x)(holds(trust\_b(d, e, x, k), s))$$
$$\wedge holds(believe(e, q\dot{\supset}y), s) \wedge entail(q, k) \supset holds(believe(d, k\theta\dot{\supset}y), s)$$

by (1), (11) and rule *CP*; used premise: {(5)}.

(12)

$$(\forall d, e, k, s, y, q)((\forall x)(holds(trust\_b(d, e, x, k), s))$$
$$\wedge holds(believe(e, q\dot{\supset}y), s) \wedge entail(q, k) \supset holds(believe(d, k\theta\dot{\supset}y), s)),$$

by (11) and rule *UG*; used premise: {(5)}.

Since (5) Axiom TR-4 is one of the axiom defined in Trust ontology, this theorem is proved.

Theorem TR-3.

$$holds(trust\_p(d, e, x, k), s) \wedge entail(q, k) \supset holds(trust\_p(d, e, x, q), s) \tag{C.19}$$

$$holds(trust\_b(d, e, x, k), s) \wedge entail(q, k) \supset holds(trust\_b(d, e, x, q), s) \tag{C.20}$$

Proof: In the following, we prove the first formula; the proof of the second formula has the same proof structure as the first has, so omitted.

(1)

$$holds(trust\_p(d, e, x, k), s) \wedge entail(q, k);$$

by rule P (for CP); used premises: {(1)}; free variables in used premises: {*d,e,x,k,q,s*}.

(2)

$$holds(trust\_p(d, e, x, k), s);$$

by rule T; used premises: {(1)}; free variables in used premises: {*d,e,x,k,s*}.

(3)

$$entail(q, k);$$

by rule T; used premises: {(1)}; free variables in used premises: {*k,q*}.

(4)

$$(\forall q')(holds(made(x, e, q'), s) \wedge entail(q', k) \supset holds(believe(d, k\dot{\supset}x), s));$$

by (2), Axiom TR-3 and rule T; used premises: {(1), Axiom TR-3}; free variables in used premises: {*d,e,x,k,s*}.

(5)

$$holds(made(x, e, c), s) \wedge entail(c, q);$$

by rule P (for CP); used premises: {(5)}; free variables in used premises: {c}.

(6)

$$holds(made(x, e, c), s) \wedge entail(c, k);$$

by (5), (3), proposition TR-2, and rule T; used premises: {(1), (5), Prop. TR-2}; free variables in used premises: {e,x,k,s,c}.

(7)

$$holds(believe(d, k \dot{\supset} x), s);$$

by (4), (6), and rule T; used premises: {(1), (5), Prop. TR-2}; free variables in used premises: {d,k,x,s}.

(8)

$$holds(believe(d, q \dot{\supset} x), s);$$

by (7), (3), proposition TR-4, and rule T; used premises: {(1), (5), Prop. TR-2, Prop. TR-4}; free variables in used premises: {d,q,x,s}.

(9)

$$holds(made(x, e, c), s) \wedge entail(c, q) \supset holds(believe(d, q \dot{\supset} x), s);$$

by (5), (8), and rule CP; used premises: {(1), Prop. TR-2, Prop. TR-4}; free variables in used premises: {d,e,x,q,s}.

(10)

$$holds(trust\_p(d, e, x, q), s);$$

by (9), Axiom TR-3, and rule T; used premises: {(1), Prop. TR-2, Prop. TR-4, Axiom TR-3}; free variables in used premises: {d,e,x,q,s}.

(11)

$$holds(trust\_p(d, e, x, k), s) \wedge entail(q, k) \supset holds(trust\_p(d, e, x, q), s);$$

by (1), (10), Axiom TR-3, and rule CP; used premises: { Prop. TR-2, Prop. TR-4, Axiom TR-3}; free variables in used premises: {}.

(12)

$$(\forall d, e, x, k, q, s)(holds(trust\_p(d, e, x, k), s) \wedge entail(q, k) \supset holds(trust\_p(d, e, x, q), s));$$

by (11) and rule UG; used premises: {Prop. TR-2, Prop. TR-4, Axiom TR-3}; free variables in used premises: {}.

Since all used premises are in defined trust ontology, we have

$$TR \vDash (\forall d, e, x, k, q, s)(holds(trust\_p(d, e, x, k), s) \wedge entail(q, k) \supset holds(trust\_p(d, e, x, q), s)).$$

The proof of the second part of t

□

Theorem TR-4 (Transitivity of trust in belief).

(a) In any situation s, if entity d trusts entity c on everything which c believes in context k, and c trusts entity e on everything which e believes in context q, then d trusts

e on everything which e believes in the conjunction of the contexts k and q.

$$(\forall x)(holds(trust\_b(d,c,x,k),s)) \wedge (\forall x)(holds(trust\_b(c,e,x,q),s)) \supset (\forall x)(holds(trust\_b(d,e,x,k\dot{\wedge}q),s)) \quad \text{(C.21)}$$

(b) In any situation s, if agent d trusts agent c on everything which c believes in context k, and c trusts agent e on everything which e performs in context q, then d trusts e on everything which e performs in the conjunction of contexts k and q.

$$(\forall x)(holds(trust\_b(d,c,x,k),s)) \wedge (\forall x)(holds(trust\_p(c,e,x,q),s)) \supset (\forall x)(holds(trust\_p(d,e,x,k\dot{\wedge}q),s)) \quad \text{(C.22)}$$

Proof of (a): In formal, we need to prove

$$TR| = (\forall d,c,e,k,q,s)((\forall x)(holds(trust\_b(d,c,x,k),s)) \wedge (\forall x)(holds(trust\_b(c,e,x,q),s)) \supset (\forall x)(holds(trust\_b(d,e,x,k\dot{\wedge}q),s))),$$

where TR denotes the set of all axioms and definitions about trust given in Chapter 5. Now we give the formal proof of this theorem as follows.

(1)

$$(\forall x)(holds(trust\_b(d,c,x,k),s)) \wedge (\forall x)(holds(trust\_b(c,e,x,q),s));$$

by rule P (for CP); used premise: $\{(1)\}$; free variables in the used premise: $\{d,\ c,\ e,\ k,\ q,\ s\}$.

(2)

$$(\forall x)(holds(trust\_b(d, c, x, k), s));$$

by (1) and rule T; used premise: $\{(1)\}$; free variables in the used premise: $\{d,\ c,\ k,\ s\}$.

(3)

$$(\forall x)(holds(trust\_b(c, e, x, q), s));$$

by (1) and rule T; used premise: $\{(1)\}$; free variables in the used premise: $\{c,\ e,\ q,\ s\}$.

(4)

$$holds(believe(e, p\dot{\supset}y), s) \wedge entail(p, k\dot{\wedge}q)$$

by rule P (for CP); used premise: $\{(4)\}$; free variables in the used premise: $\{p,y\}$.

(5)

$$holds(trust\_b(d, c, x', k), s));$$

by (2) and rule US; used premise: $\{(1)\}$; free variables in the used premise: $\{d,\ c,\ k,\ s\}$.

(6)

$$holds(trust\_b(c, e, x', q), s);$$

by (3) and rule US; used premise: $\{(1)\}$; free variables in the used premise: $\{c,\ e,\ q,\ s\}$.

(7)

$$holds(trust\_b(d, c, x', k\dot{\wedge}q), s);$$

by (5), Proposition TR-1, theorem TR-3, and rule T; used premises: {(1), prop. TR-1, theorem TR-3}; free variables in used premises: $\{d,\ c,\ k,\ q,\ s\}$.

(8)

$$holds(trust\_b(c, e, x', k\dot{\wedge}q), s);$$

by (6), Proposition TR-1, theorem TR-3, and rule T; used premises: {(1), prop. TR-1, theorem TR-3}; free variables in used premises: {*c, e, k, q, s*}.

(9)

$$(\forall x)holds(trust\_b(d, c, x, k\dot{\wedge}q), s);$$

by (7), and rule UG; used premises: {(1), prop. TR-1, theorem TR-3}; free variables in used premises: {*d, c, k, q, s*}.

(10)

$$(\forall x)holds(trust\_b(c, e, x, k\dot{\wedge}q), s);$$

by (8), and rule UG; used premises: {(1), prop. TR-1, theorem TR-3}; free variables in used premises: {*c, e, k, q, s*}.

(11)

$$holds(believe(c, k\dot{\wedge}q\dot{\supset}y), s);$$

by (4), (10), Theorem TR-2, and rule T ; used premises: {(1), (4), prop. TR-1, theorem TR-3, theorem TR-2}; free variables in the used premise: {*c, k, q, y, s*}.

(12)

$$entail(k\dot{\wedge}q, k\dot{\wedge}q)$$

by definition of *entail* (def. TR-2) and rule T; used premises: {def. TR-2}.

(13)

$$holds(believe(d, k\dot{\wedge}q\dot{\supset}y), s);$$

by (9), (11), (12), Theorem TR-2, and rule T ; used premises: {(1), (4), prop. TR-1, theorem TR-3, theorem TR-2, def. TR-2}; free variables in the used premises: {*d, k, q, y, s*}.

(14)

$$holds(believe(e, p\dot{\supset}y), s) \wedge entail(p, k\dot{\wedge}q) \supset holds(believe(d, k\dot{\wedge}q\dot{\supset}y), s);$$

by (4), (13), and rule *CP*; used premises: {(1), prop. TR-1, theorem TR-3, theorem TR-2, def. TR-2}; free variables in the used premises: {*d, e, k, q, s*}.

(15)

$$(\forall p)(holds(believe(e, p\dot{\supset}y), s) \wedge entail(p, k\dot{\wedge}q) \supset holds(believe(d, k\dot{\wedge}q\dot{\supset}y), s));$$

by (14) and rule *UG*; used premises: {(1), prop. TR-1, theorem TR-3, theorem TR-2, def. TR-2}; free variables in the used premises: {*d, e, k, q, s*}.

(16)

$$holds(trust\_b(d, e, y, k\dot{\wedge}q), s));$$

by (15), axiom TR-4 and rule *T*; used premises: {(1), prop. TR-1, theorem TR-3, theorem TR-2, def. TR-2, axiom TR-4}; free variables in the used premises: {*d, e, k, q, s*}.

(17)

$$(\forall x)(holds(trust\_b(d, e, x, k\dot{\wedge}q), s));$$

by (16) and rule *UG*; used premises: {(1), prop. TR-1, theorem TR-3, theorem TR-2, def. TR-2, axiom TR-4}; free variables in the used premises: {*d, e, k, q, s*}.

(18)

$$(\forall x)(holds(trust\_b(d, c, x, k), s)) \wedge (\forall x)(holds(trust\_b(c, e, x, q), s)) \supset (\forall x)(holds(trust\_b(d, e, x, k\dot{\wedge}q), s));$$

by (1), (17) and rule *UG*; used premises: {prop. TR-1, theorem TR-3, theorem TR-2, def. TR-2, axiom TR-4}; free variables in the used premises: {}.

(19)

$$(\forall d, c, e, k, q, s)((\forall x)(holds(trust\_b(d, c, x, k), s)) \wedge (\forall x)(holds(trust\_b(c, e, x, q), s)) \supset (\forall x)(holds(trust\_b(d, e, x, k\dot{\wedge}q), s)));$$

by (18) and rule *UG*; used premises: {prop. TR-1, theorem TR-3, theorem TR-2, def. TR-2, axiom TR-4}; free variables in the used premises: {}.

Since all premises used are in the defined Trust ontology, Theorem TR-4 (a) is proved.

Proof of (b): we need to prove

$$TR| = (\forall d, c, e, k, q, s)((\forall x)(holds(trust\_b(d, c, x, k), s)) \wedge (\forall x)(holds(trust\_p(c, e, x, q), s)) \supset (\forall x)(holds(trust\_p(d, e, x, k\dot{\wedge}q), s))),$$

(1)

$$(\forall x)(holds(trust\_b(d, c, x, k), s)) \wedge (\forall x)(holds(trust\_p(c, e, x, q), s));$$

by rule *P* (for *CP*); used premise: {(1)}; free variables in the used premise: {*d, c, e, k, q, s*}.

(2)

$$(\forall x)(holds(trust\_b(d, c, x, k), s));$$

by (1) and rule *T*; used premise: {(1)}; free variables in the used premise: {*d, c, k, s*}.

(3)

$$(\forall x)(holds(trust\_p(c, e, x, q), s));$$

by (1) and rule T; used premise: $\{(1)\}$; free variables in the used premise: $\{c,\ e,\ q,\ s\}$.

(4)

$$holds(made(y, e, p), s) \wedge entail(p, k\dot{\wedge}q)$$

by rule P (for CP); used premise: $\{(4)\}$; free variables in the used premise: $\{p,y\}$.

(5)

$$holds(trust\_b(d, c, x', k), s));$$

by (2) and rule US; used premise: $\{(1)\}$; free variables in the used premise: $\{d,\ c,\ k,\ s\}$.

(6)

$$holds(trust\_p(c, e, x', q), s);$$

by (3) and rule US; used premise: $\{(1)\}$; free variables in the used premise: $\{c,\ e,\ q,\ s\}$.

(7)

$$holds(trust\_b(d, c, x', k\dot{\wedge}q), s);$$

by (5), Proposition TR-1, theorem TR-3, and rule T; used premises: {(1), prop. TR-1, theorem TR-3}; free variables in used premises: $\{d,\ c,\ k,\ q,\ s\}$.

(8)

$$holds(trust\_p(c, e, x', k\dot{\wedge}q), s);$$

by (6), Proposition TR-1, theorem TR-3, and rule T; used premises: {(1), prop. TR-1, theorem TR-3}; free variables in used premises: $\{c,\ e,\ k,\ q,\ s\}$.

(9)

$$(\forall x)holds(trust\_b(d, c, x, k\dot{\wedge}q), s);$$

by (7), and rule *UG*; used premises: {(1), prop. TR-1, theorem TR-3}; free variables in used premises: {*d, c, k, q, s*}.

(10)

$$(\forall x)holds(trust\_p(c, e, x, k\dot{\wedge}q), s);$$

by (8), and rule *UG*; used premises: {(1), prop. TR-1, theorem TR-3}; free variables in used premises: {*c, e, k, q, s*}.

(11)

$$holds(believe(c, k\dot{\wedge}q\dot{\supset}y), s);$$

by (4), (10), Theorem TR-1, and rule *T* ; used premises: {(1), (4), prop. TR-1, theorem TR-3, theorem TR-1}; free variables in the used premise: {*c, k, q, y, s*}.

(12)

$$entail(k\dot{\wedge}q, k\dot{\wedge}q)$$

by definition of *entail* (def. TR-2) and rule *T*; used premises: {def. TR-2}.

(13)

$$holds(believe(d, k\dot{\wedge}q\dot{\supset}y), s);$$

by (9), (11), (12), Theorem TR-2, and rule *T* ; used premises: {(1), (4), prop. TR-1, theorem TR-3, theorem TR-1, def. TR-2, theorem TR-2}; free variables in the used premises: {*d, k, q, y, s*}.

(14)

$$holds(made(y, e, p), s) \wedge entail(p, k\dot{\wedge}q) \supset holds(believe(d, k\dot{\wedge}q\dot{\supset}y), s);$$

by (4), (13), and rule *CP*; used premises: {(1), prop. TR-1, theorem TR-3, theorem TR-1, def. TR-2, theorem TR-2}; free variables in the used premises: {*d, e, k, q, s*}.

(15)

$$(\forall p)(holds(made(y, e, p), s) \wedge entail(p, k\dot{\wedge}q) \supset holds(believe(d, k\dot{\wedge}q\dot{\supset}y), s));$$

by (14) and rule *UG*; used premise: {(1), prop. TR-1, theorem TR-3, def. TR-2, theorem TR-1}; free variables in the used premises: {*d, e, k, q, s*}.

(16)

$$holds(trust\_p(d, e, y, k\dot{\wedge}q), s));$$

by (15), axiom TR-3 and rule *T*; used premises: {(1), prop. TR-1, theorem TR-3, theorem TR-1, def. TR-2, theorem TR-2, axiom TR-3}; free variables in the used premises: {*d, e, k, q, s*}.

(17)

$$(\forall x)(holds(trust\_p(d, e, x, k\dot{\wedge}q), s));$$

by (16) and rule *UG*; used premises: {(1), prop. TR-1, theorem TR-3, theorem TR-1, def. TR-2, theorem TR-2, axiom TR-3}; free variables in the used premises: {*d, e, k, q, s*}.

(18)

$$(\forall x)(holds(trust\_b(d, c, x, k), s)) \wedge (\forall x)(holds(trust\_p(c, e, x, q), s)) \supset (\forall x)(holds(trust\_p(d, e, x, k\dot{\wedge}q), s));$$

by (1), (17) and rule *UG*; used premises: {prop. TR-1, theorem TR-3, theorem TR-1, def. TR-2, theorem TR-2, axiom TR-3}; free variables in the used premises: {}.

(19)

$$(\forall d, c, e, k, q, s)((\forall x)(holds(trust\_b(d, c, x, k), s)) \wedge (\forall x)(holds(trust\_p(c, e, x, q), s)) \supset (\forall x)(holds(trust\_p(d, e, x, k\dot{\wedge}q), s)));$$

by (18) and rule *UG*; used premises: {prop. TR-1, theorem TR-3, theorem TR-1, def. TR-2, theorem TR-2, axiom TR-3}; free variables in the used premises: {}.

Since all premises used are in the defined Trust ontology, Theorem TR-4 (b) is proved. □

Theorem TR-5: For every trust path in a trust network, the corresponding trust relationship is valid by the trust ontology.

Proof. Assume a trust path is

$$(e_0, e_1, e_2, ..., e_{n-1}, e_n),$$

and the labels on the arcs in order are:

$$(b, k_0), (b, k_1), ..., (b, k_{n-2}), (t, k_{n-1}),$$

where t may be b or p. Assume t is p. The proof for t=b is similar. This trust path represents the following trust relationship that holds in any situation after the trust path is obtained,

$$\forall x, holds(trust\_p(e_0, e_n, x, k_0 \wedge k_1 \wedge ... \wedge k_{n-1}), s). \quad \text{(C.23)}$$

Now, we prove this trust relationship is valid. From the above assumption and the

definition of trust networks, we have the following inter-individual trust relationships:

$$\forall x, holds(has\_b\_tr(e_0, e_1, x, k_0), s),$$

$$\forall x, holds(has\_b\_tr(e_1, e_2, x, k_1), s),$$

$$...$$

$$\forall x, holds(has\_b\_tr(e_{n-2}, e_{n-1}, x, k_{n-2}), s),$$

$$\forall x, holds(has\_p\_tr(e_{n-1}, e_n, x, k_{n-1}), s).$$

Consider

$$entail(k_0 \dot{\wedge} k_1 \dot{\wedge} ... \dot{\wedge} k_{n-1}, k_0),$$

$$...,$$

$$entail(k_0 \dot{\wedge} k_1 \dot{\wedge} ... \dot{\wedge} k_{n-1}, k_{n-1}).$$

Apply axiom TR-5 and theorem TR-3 to all above sentences, we have,

$$\forall x, holds(trust\_b(e_0, e_1, x, k_0 \dot{\wedge} k_1 \dot{\wedge} ... \dot{\wedge} k_{n-1}), s),$$

$$\forall x, holds(trust\_b(e_1, e_2, x, k_0 \dot{\wedge} k_1 \dot{\wedge} ... \dot{\wedge} k_{n-1}), s),$$

$$...$$

$$\forall x, holds(trust\_b(e_{n-2}, e_{n-1}, x, k_0 \dot{\wedge} k_1 \dot{\wedge} ... \dot{\wedge} k_{n-1}), s),$$

$$\forall x, holds(trust\_p(e_{n-1}, e_n, x, k_0 \dot{\wedge} k_1 \dot{\wedge} ... \dot{\wedge} k_{n-1}), s).$$

Repeat applying Theorem TR-4(a) to the above sentences from the last pair till the first, then we have (C.23). Therefore, this trust relationship is valid.

□

Theorem TR-6: For every valid trust relationship by the trust ontology, there exists at least a trust path within the context of the trust and with finite length in a trust network.

Proof. Assume a trust relationship holds at situation S, i.e. it is valid.

$$\forall x, holds(trust\_p(Trustor, Trustee, x, K), S) \tag{C.24}$$

where $Trustor \neq Trustee$.

From the causal completeness assumption, this trust relationship must be derived from axiom TR-5, theorems TR-3 and TR-4 with a finite number of steps. Consider Theorem TR-3 actually just apply the same trust to a stricter context. This trust relationship must be derived either from an direct inter-individual trust relationship or by Theorem TR-4 via a sequence of intermediate entities.

Now, we prove this theorem by applying strong mathematical induction to the minimum number (denoted as i) of intermediate entities through which *Trustor* trusts *Trustee*.

First, prove that the theorem is true in the basic cases i = 0 and i = 1.

(i) i = 0, i.e. there is no intermediate entities. From causal completeness assumption, the assumed trust relationship must be derived from direct trust by using axiom TR-5 and possibly Theorem TR-3 from an inter-individual relationship like

$$\forall x, holds(has\_p\_tr(Trustor, Trustee, x, Q), S),$$

and

$$entail(Q, K).$$

In this case, according to the definition of trust networks, arc $(Trustor, Trustee, p, Q) \in A$. This single arc is the trust path within context K.

(ii) i = 1, this is the case of derived trust relationship via one intermediate entity. Assume that there is an entity e, such that the assumed trust relationship (C.24) is derived by applying theorems 8 (5.8) and 7 (C.19)from

$$\forall x, holds(trust\_b(Trustor, e, x, Q), S),$$

$$entail(Q, K),$$

$$\forall x, holds(trust\_p(e, Trustee, x, R), S),$$

$$entail(R, K)$$

Here, $e \neq Trustor$ and $e \neq Trustee$

Since we assume i=1, the trust relationship between *Trustor* and e, and trust relationship between e and *Trustee* must be direct trust. In this case, according to case (i), arcs $(Trustor, e)$ and $(e, Trustee)$ must be in the trust network, thus we have a trust path within context K without circle and with length 2: $(Trustor, e, Trustee)$.

(iii) secondly, assume that this theorem is true when $i \leq n$, n is any positive integer. Now we prove that the theorem is true when $i = n + 1$, i.e. *Trustor* trusts *Trustee* via $n+1$ intermediate entities. We only need to consider $i > 1$ (case of $i = 1$ has been proved in (ii) above).

In this case, the assumed trust relationship (C.24) must be derived by applying theorems 4 (5.26) and/or 3(C.19) via more than one entity. Assume e is the last intermediate entity used to derive (C.24). This means the assumed trust relationship (C.24) is derived from

$$\forall x, holds(trust\_b(Trustor, e, x, Q), S),$$

$$entail(Q, K),$$

$$\forall x, holds(trust\_p(e, Trustee, x, R), S),$$

$$entail(R, K),$$

Because e is one of the $n+1$ intermediate entities, the minimum numbers of intermediate entities between *Trustor* and e and between e and *Trustee* must be less than or equal to n. According to our assumption, there is a trust path within context K without circle and with limited length from *Trustor* to e: $(Trustor, ..., e)$. Similarly, there is also a trust path within context K without circle and with limited length from e to *Trustee*, *(e, ...,Trustee)*. Therefore, there is a trust path within context K without circle and with finite length from *Trustor* to *Trustee*, *(Trustor,..., e,...,Trustee)*.

Therefore, for any finite number, the theorem is true.

□

Appendix D

Example: Output from KP Reasoner

The following is the output from KP reasoner for the application example given in Chapter 3.

Load proposition

http://www.eil.utoronto.ca/kp/ex/arctic/polarbears.html#EndangeredPolarBears

Solve: Derived Proposition

http://www.eil.utoronto.ca/kp/ex/arctic/polarbears.html#EndangeredPolarBears

=> Author: Andrew Derocher

=> Publisher: CBC

Information creator: Andrew Derocher is trusted.

This proposition is believed.

..Load support proposition

http://www.eil.utoronto.ca/kp/ex/arctic/polarbears.html#MeltingArcticSeaIce

Pause solving, to solve support proposition(s) first.

Solve: Equivalent Proposition

http://www.eil.utoronto.ca/kp/ex/arctic/polarbears.html#MeltingArcticSeaIce

..Load support proposition

http://www.eil.utoronto.ca/kp/ex/arctic/arcticseaice.html#MeltingArcticSeaIce

Pause solving, to solve the support proposition first.

Solve: Asserted Proposition

http://www.eil.utoronto.ca/kp/ex/arctic/arcticseaice.html#MeltingArcticSeaIce

=> Publisher: NASA

Information creator: NASA is trusted.

This proposition is believed.

=> Believed truth value: True

Solve: Equivalent Proposition

http://www.eil.utoronto.ca/kp/ex/arctic/polarbears.html#MeltingArcticSeaIce

Dependent on:

http://www.eil.utoronto.ca/kp/ex/arctic/arcticseaice.html#MeltingArcticSeaIce

Whose believed truth value: True

Two propositions have the same content.

=> Believed truth value: True

Solve: Derived Proposition

http://www.eil.utoronto.ca/kp/ex/arctic/polarbears.html#EndangeredPolarBears

This proposition is believed.

Dependent on:

http://www.eil.utoronto.ca/kp/ex/arctic/polarbears.html#MeltingArcticSeaIce

Whose believed truth value: True

=>Believed truth value: True

Appendix E

Example: KP Annotation in Finance

This appendix demonstrates KP annotation for the sample investment newsletter given in section 9.2.1 of Chapter 9.

Recommendation: Buy

The first kp-proposition is the recommendation to buy. This is a derived proposition, which depends on several arguments. The proposition is annotated as follows.

<kp:DerivedProp rdf:id="#RecommToBuy"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "#Argument-pos-1"

kp:isDependentOn = "#Argument-pos-2"

kp:isDependentOn = "#Argument-pos-3"

kp:isDependentOn = "#Argument-pos-4"

kp:isDependentOn = "#Argument-neg-1"

kp:inField = "investment advices"

>

Recommendation: Buy

</kp:DerivedProp>

We will give the supporting propositions for the above derived proposition later.

Target Price: $94.13

The second proposition is an assertion about the target price of the recommended company.

<kp:AssertedProp rdf:id="#TargetPrice"
kp:author = "http://w-finance.com/ w/"
kp:inField = "investment advices"
>
Target Price: $94.13
</kp:AssertedProp>

Three equivalent propositions about the current price, 52 week high and 52 week low prices of LMT are cited from NYSE on 15 November 2006

Current Price (2006-11-15): 88.54

<kp:EquivalentProp rdf:id="#CurrentPrice"
kp:isDependentOn = "http://www.nyse.com/about/listed/lcddata.html?ticker=LMT#CurPr"
>
Current Price (2006-11-10): 85.75
</kp:EquivalentProp>

52wk High (2006-10-24): 89.89

<kp:EquivalentProp rdf:id="#52wkHigh"
kp:isDependentOn = "http://www.nyse.com/about/listed/lcddata.html?ticker=LMT#52wkHigh"
>
52wk High (2006-10-24): 89.89
</kp:EquivalentProp>

52wk Low (2005-11-15): 59.55

<kp:EquivalentProp rdf:id="#52wkLow"

kp:isDependentOn = "http://www.nyse.com/about/listed/lcddata.html?ticker=LMT#52wkHigh"

>

52wk Low (2005-11-10): 59.17

</kp:EquivalentProp>

Key Positive Arguments :

Four positive arguments are annotated as follows.

<kp:DerivedProp rdf:id="#Argument-pos-1"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "#LMT-NetSale06Q3"

kp:isDependentOn = "#LMT-NetSale05Q3"

kp:isDependentOn = "#LMT-OperatingProfit06Q3"

kp:isDependentOn = "#LMT-OperatingProfit05Q3"

kp:isDependentOn = "#LMT-CostOfSale06Q3"

kp:isDependentOn = "#LMT-CostOfSale05Q3"

kp:inField = "financial analysis"

>

Expansion of margins: Gross margin [1] *was up 1.67 point to 8.36% from 6.69% in the year-earlier period; operating marginoperating margin = operating income / net sale was up 1.71 to 9.42% from 7.67%.*

</kp:DerivedProp>

<kp:DerivedProp rdf:id="#Argument-pos-2"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "#LMT-TotalAssets06Q3"

kp:isDependentOn = "#LMT-TotalRecentLiabilities06Q3"

[1] gross margin = (net sale - cost of sale) / net sale

kp:isDependentOn = "#LMT-Euity06Q3"

kp:isDependentOn = "#LMT-TotalAssets05"

kp:isDependentOn = "#LMT-TotalRecentLiabilities05"

kp:isDependentOn = "#LMT-Euity05"

kp:isDependentOn = "#LMT-CashFlows06"

kp:isDependentOn = "#LMT-CashFlows05"

kp:inField = "financial analysis"

>

Strong free cash flow and improving balance sheet;

</kp:DerivedProp>

<kp:DerivedProp rdf:id="#Argument-pos-3"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "#LMT-NetIncome06Q3"

kp:isDependentOn = "#LMT-NetIncome05Q3"

kp:isDependentOn = "#LMT-EarningsPerShare06Q3"

kp:isDependentOn = "#LMT-EarningsPerShare05Q3"

kp:isDependentOn = "#LMT-EffectiveTaxRate06Q3"

kp:isDependentOn = "#LMT-EffectiveTaxRate05Q3"

kp:inField = "financial analysis"

>

Big increase of profit: Third-quarter net income rose 47% to $629 million, or $1.46 a share, from $427 million, or 96 cents a share, in the year-earlier period. Its tax rate dropped to 22.8% in the quarter from 30.3% a year ago.

</kp:DerivedProp>

<kp:DerivedProp rdf:id="#Argument-pos-4"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "#Event-LMT-Orion"

kp:inField = "financial analysis"

>

Won major contracts, for example Orion project.

</kp:DerivedProp>

Key Negative Arguments :

<kp:DerivedProp rdf:id="#Argument-neg-1"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "#Event-Dems-06Election"

kp:isDependentOn = "#Event-Rumsfeld"

kp:inField = "financial analysis"

>

Uncertainty in defense policy.

</kp:DerivedProp>

Recent Events :

The presented recent events are annotated as follows.

<kp:DerivedProp rdf:id="#Event-Dems"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "http://www.cnn.com/2006/POLITICS

/11/08/election.house/index.html#DemsWinHouse"

kp:isDependentOn = "http://www.cnn.com/2006/POLITICS

/11/08/election.house/index.html#DemsCnrlSenate"

kp:isDependentOn = "http://www.cnn.com/2006/POLITICS

/11/08/election.house/index.html#DemsMajorGov"

kp:isDependentOn = "http://www.marketwatch.com/News/Story

/Story.aspx?guid=%7BB739E9DE%2D99F7%2D475D%2D9B5B

%2D7BEBB9525587%7D#TradHiber"

kp:inField = "news & comments"

>

Democrats wins 2006 midterm Elections.

</kp:DerivedProp>

<kp:DerivedProp rdf:id="#Event-Rumsfeld"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "http://www.cnn.com/2006/POLITICS

/11/08/rumsfeld.ap/index.html#Rumsfeld"

kp:inField = "news & comments"

>

On November 8, 2006, Defense Secretary Donald H. Rumsfeld stepped down, which clouds defense policy. Defense Secretary Donald Rumsfeld's swift exit Wednesday from the Pentagon in the wake of the Republican Party's midterm election rout removes a lightning rod for criticism plaguing the Bush administration. But it also raises questions about the direction of the nation's defense spending.

</kp:DerivedProp>

<kp:DerivedProp rdf:id="#Event-LMT-Orion"

kp:author = "http://w-finance.com/ w/"

kp:isDependentOn = "http://www.nasa.gov/home/hqnews

/2006/aug/HQ_06305_Orion_contract.html#Orion-LMT"

kp:inField = "news & comments"

>

LMT won contract on Orion space program. On August 31, 2006, NASA selected Thursday Lockheed Martin Corp. as the prime contractor to design, develop, and build Orion, America's spacecraft for a new generation of explorers. The estimated

value for the project is $3.9 billion.

</kp:DerivedProp>

Revenue Highlights :

In this part, several XBRL items are specified to support the earlier arguments.

<kp-br:XBRL_Item rdf:id = "#LMT-NetIncome06Q3"

kp:inDoc="http://www.sec.gov/Archives/edgar/data

/936468/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:NetIncome"

kp-br:value = "629000000"

kp-br:unit = "USD"

kp-br:startDate = 2006-07-01

kp-br:endDate = 2006-09-30

kp-br:entity = "Lockheed Martin Corporation">

$629 million </kp-br:XBRL_Item>

<kp-br:XBRL_Item rdf:id = "#LMT-EarningsPerShare06Q3"

kp:inDoc="http://www.sec.gov/Archives/edgar/data

/936468/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:DilutedEarningsPerShareNetIncome"

kp-br:value = "1.46"

kp-br:unit = "decimal"

kp-br:startDate = 2006-07-01

kp-br:endDate = 2006-09-30

kp-br:entity = "Lockheed Martin Corporation">

$1.46 </kp-br:XBRL_Item>

<kp-br:XBRL_Item rdf:id = "#LMT-NetIncome05Q3"

kp:inDoc="http://www.sec.gov/Archives/edgar/data

/936468/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:NetIncome"

kp-br:value = "427000000"

kp-br:unit = "USD"

kp-br:startDate = 2005-07-01

kp-br:endDate = 2005-09-30

kp-br:entity = "Lockheed Martin Corporation"/>

<kp-br:XBRL_Item rdf:id = "#LMT-EarningsPerShare05Q3"

kp:inDoc="http://www.sec.gov/Archives/edgar/data

/936468/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:DilutedEarningsPerShareNetIncome"

kp-br:value = "0.96"

kp-br:unit = "decimal"

kp-br:startDate = 2005-07-01

kp-br:endDate = 2005-09-30

kp-br:entity = "Lockheed Martin Corporation">

$0.96 </kp-br:XBRL_Item>

Margins Highlights :

The concepts of margins are defined as follows:

net margin = net income / net sale

gross margin = (net sale - cost of sale) / net sale

operating margin = operating income / net sale

<kp-br:XBRL_Item rdf:id = "#LMT-NetSale06Q3"
kp:inDoc="http://www.sec.gov/Archives/edgar/data
/936468/000119312506219311/lmt-20060930.xml"
kp:inField = "financial statement"
kp-br:item = "usfr-pte:OperatingRevenue"
kp-br:value = "9605000000"
kp-br:unit = "USD"
kp-br:startDate = 2006-07-01
kp-br:endDate = 2006-09-30
kp-br:entity = "Lockheed Martin Corporation"/>

<kp-br:XBRL_Item rdf:id = "#LMT-NetSale05Q3"
kp:inDoc="http://www.sec.gov/Archives/edgar/data
/936468/000119312506219311/lmt-20060930.xml"
kp:inField = "financial statement"
kp-br:item = "usfr-pte:OperatingRevenue"
kp-br:value = "9201000000"
kp-br:unit = "USD"
kp-br:startDate = 2006-07-01
kp-br:endDate = 2006-09-30
kp-br:entity = "Lockheed Martin Corporation"/>

<kp-br:XBRL_Item rdf:id = "#LMT-OperatingProfit06Q3"
kp:inDoc="http://www.sec.gov/Archives/edgar/data
/936468/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:OperatingProfit"

kp-br:value = "905000000"

kp-br:unit = "USD"

kp-br:startDate = 2006-07-01

kp-br:endDate = 2006-09-30

kp-br:entity = "Lockheed Martin Corporation"/>

<kp-br:XBRL_Item rdf:id = "#LMT-OperatingProfit05Q3"

kp:inDoc="http://www.sec.gov/Archives/edgar/data

/936468/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:OperatingProfit"

kp-br:value = "706000000"

kp-br:unit = "USD"

kp-br:startDate = 2005-07-01

kp-br:endDate = 2005-09-30

kp-br:entity = "Lockheed Martin Corporation"/>

<kp-br:XBRL_Item rdf:id = "#LMT-CostOfSale06Q3"

kp:inDoc="http://www.sec.gov/Archives/edgar/data/936468

/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:CostGoodsServicesSold"

kp-br:value = "8802000000"

kp-br:unit = "USD"

kp-br:startDate = 2006-07-01

kp-br:endDate = 2006-09-30

kp-br:entity = "Lockheed Martin Corporation"/>

<kp-br:XBRL_Item rdf:id = "#LMT-CostofSale05Q3"

kp:inDoc="http://www.sec.gov/Archives/edgar/data/936468

/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:CostGoodsServicesSold"

kp-br:value = "8585000000"

kp-br:unit = "USD"

kp-br:startDate = 2006-07-01

kp-br:endDate = 2006-09-30

kp-br:entity = "Lockheed Martin Corporation"/>

Cash Flows Highlight :

Cash from operations (year to date) rose 9.9% to **$3,450 million** from **$3,138 million**.

<kp-br:XBRL_Item rdf:id = "#LMT-CashFlows06"

kp:inDoc="http://www.sec.gov/Archives/edgar/data/936468

/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:NetCashFlowsProvidedUsedOperatingActivities"

kp-br:value = "3450000000"

kp-br:unit = "USD"

kp-br:startDate = 2006-01-01

kp-br:endDate = 2006-09-30

kp-br:entity = "Lockheed Martin Corporation">

$3,450 million

</kp-br:XBRL_DataItem>

<kp-br:XBRL_Item rdf:id = "#LMT-CashFlows05"

kp:inDoc="http://www.sec.gov/Archives/edgar/data/936468

/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item="usfr-pte:NetCashFlowsProvidedUsedOperatingActivities"

kp-br:value = "31380000000"

kp-br:unit = "USD"

kp-br:startDate = "2005-01-01"

kp-br:endDate = "2005-09-30"

kp-br:entity = "Lockheed Martin Corporation">

$3,138 million

</kp-br:XBRL_DataItem>

Balance Sheet Highlights :

<kp-br:XBRL_Item rdf:id = "#LMT-TotalAssets"

kp:inDoc="http://www.sec.gov/Archives/edgar/data/936468

/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:Assets"

kp-br:value = "29093000000"

kp-br:unit = "USD"

kp-br:atDate = 2006-09-30

kp-br:entity = "Lockheed Martin Corporation"> $29,093

</kp-br:XBRL_DataItem>

<kp-br:XBRL_Item rdf:id = "#LMT-TotalRecentLiabilities"

kp:inDoc="http://www.sec.gov/Archives/edgar/data/936468

/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:CurrentLiabilities"

kp-br:value = "10383000000"

kp-br:unit = "USD"

kp-br:atDate = "2006-09-30"

kp-br:entity = "Lockheed Martin Corporation">

$10,383

</kp-br:XBRL_DataItem>

<kp-br:XBRL_Item rdf:id = "#LMT-Equity"

kp:inDoc="http://www.sec.gov/Archives/edgar/data/936468

/000119312506219311/lmt-20060930.xml"

kp:inField = "financial statement"

kp-br:item = "usfr-pte:StockholdersEquity"

kp-br:value = "8083000000"

kp-br:unit = "USD"

kp-br:atDate = 2006-09-30

kp-br:entity = "Lockheed Martin Corporation">

$8,083

</kp-br:XBRL_DataItem>

Bibliography

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of 33rd Hawaii International Conference on System Sciences*, 2000.

[2] E. Adams. *A Primer of Probability Logic*. CSLI, Stanford University, 1998.

[3] J. Alexander and M. Tate. *Web Wisdom: how to evaluate and create information quality on the web*. Lawrence Erlbaum Associates Publishers, 1999.

[4] J. Allen and G. Ferguson. Actions and events in interval temporal logic. *J. Logic and Computation*, 4(5):531–579, 1994.

[5] J. F. Allen. Time and time again: The many ways to represent time. *International Journal of Intelligent Systems*, 6(4), 1991.

[6] A.S.Rao and M.P.Georgeff. Modeling rational agents within a bdi-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann, 1991.

[7] B. Barber. *The logic and limits of trust*. Rutgers University Press, New Brunswick, N.J., 1983.

[8] C. L. Barry. User-defined relevance criteria: An exporatory study. *Journal Of the American Society for Information Science*, 45(3):149–159, 1994.

[9] C. L. Barry and L. Schamber. Users' criteria for relevance evaluation: A cross-situational comparison. *Information Processing & Management*, 34(2/3):219–236, 1998.

[10] T. Berners-Lee. Semantic web road map. 1998.

[11] T. Berners-Lee. Semantic web status and direction. In *2003 International Semantic Web Conference*, October 2003.

[12] T. Berners-Lee. Putting the web back into semantic web. In *2005 International Semantic Web Conference*, November 2005.

[13] T. Berners-Lee, W. Hall, J. A. Hendler, K. O'hara, N. Shadbolt, and D. J. Weitzner. A framework for web science. *Foundations and Trends in Web Science*, 1(1):1–130, 2006.

[14] T. Berners-Lee, J. A. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):35–43, May 2001.

[15] R. Bhatnager and L. Kanal. Handling uncertain information: A review of numeric and non-numeric methods. 1986.

[16] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The keynote trust-management system (version 2). In *http://www.crypto.com/papers/rfc2704.txt*, 1999.

[17] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, 1996.

[18] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance-checking in the policymaker trust-management system. In *Proc. 2nd Conference on Financial Cryptography, Anguilla, LNCS 1465*, pages 251–265. Springer-Verlag, 1998.

[19] K. Blomqvist. The many faces of trust. *Scandinavian Journal of Management*, 13(3):271–286, 1997.

[20] R. Brachman and H. J. Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann, 2003.

[21] M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[22] P. Buneman, S. Khanna, and W. Tan. Data provenance: Some basic issues. In *Foundations of Software Technology and Theoretical Computer Science*, 2000.

[23] P. Buneman, S. Khanna, and W. Tan. Why and where: A characterization of data provenance. In *Proceedings of International Conference on Database Theory (ICDT)*, 2001.

[24] S. Buvac and I. Mason. Propositional logic of context. In *Proceedings of AAAI'1993*, 1993.

[25] P. Calvert. Uk government says xbrl will be mandatory for company tax filings from 2010. 3 2006.

[26] P. Charter. Public-key infrastructure (x.509)(pkix).

[27] B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, 1980.

[28] Y.-H. Chu. *Trust Management for the World Wide Web*. Master Thesis, MIT, 1997.

[29] J. Coleman. *Foundations of Social Theory*. Harvard University Press, 1990.

[30] C. H. Coombs, R. M. Dawes, and A. Tversky. *Mathematical Psychology*. Prentical-Hall, Inc., 1970.

[31] Cornell-University-Library. Evaluating web sites: Criteria and tools. In *http://www.library.cornell.edu/olinuris/ref/research/webeval.html*, 2002.

[32] C. Cox. Speech by sec chairman: Opening remarks to the practising law institute's sec speaks series. 3 2006.

[33] M. Creswell. Modal logic. 2001.

[34] J. de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28(2):127–162, 1986.

[35] J. de Kleer, K. Forbus, and D. McAllester. Truth maintenance systems (tutorial sa5). In *IJCAI-89, SA5*, page 182 225, 1989.

[36] J. de Kleer, K. Forbus, and D. McAllester. Truth maintenance systems (tutorial sa5). In *IJCAI-89, SA5*, page 227 279, 1989.

[37] R. Demolombe. To trust information sources: a proposal for a modal logical framework. In C. Castelfranchi and Y.-H. Tan, editors, *Trust and deception in virtual societies*, pages 111–124. Kluwer Academic Publishers, 2001.

[38] M. Deutsch. Cooperation and trust: Some theoretical notes. In M. Jones, editor, *Nebraska Symposium on Motivation*, volume X, pages 275–318, 1962.

[39] M. Deutsch. *The Resolution of Conflict*. Yale University Press, New Haven and London, 1973.

[40] M. Deutsch. Trust and suspicion: Theoretical notes. In *The Resolution of Conflict*, pages 143–176, New Haven and London, 1973. Yale University Press.

[41] L. Ding, P. Kolari, S. G. Finin, A. Joshi, Y. Peng, and Y. Yesha. Modeling and evaluating trust network inference. In *The Seventh International Workshop on Trust in Agent Societies, at AAMAS 2004*, 2005.

[42] L. Ding, P. Kolari, T. Finin, and A. Joshi. On homeland security and the semantic web: A provenance and trust aware inference framework. In *Proceedings of the AAAI Spring Symposium on AI Technologies for Homeland Security*, 2005.

[43] R. M. D. W. Dodds, P.S. An experimental study of search in global social networks. *Science*, 301(8):827–829, 2003.

[44] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[45] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[46] R. Falcone and C. Castelfranchi. Trust dynamics: How trust is influenced by direct experiences and by trust itself. In *AAMAS'04*, July 2004.

[47] R. Fikes, M. Cutkosky, T. R. Gruber, and J. V. Baalen. Knowledge sharing technology project overview. In *Knowledge Systems Laboratory, Stanford University*, 1991.

[48] I. Foster and C. Kesselman. *The grid : blueprint for a new computing infrastructure*. Elsevier, 2004.

[49] M. S. Fox. Enterprise modeling. *AI Magazine*, pages 109–121, 1998.

[50] M. S. Fox, M. Barbuceanu, M. Gruninger, and J. Lin. An organisation ontology for enterprise modeling. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*, pages 131–152. AAAI/MIT Press, 1998.

[51] M. S. Fox and J. Huang. Knowledge provenance: An approach to modeling and maintaining the evolution and validity of knowledge. *http://www.eil.utoronto.ca/km/papers/fox-kp1.pdf*, 2003.

[52] M. S. Fox and J. Huang. Knowledge provenance in enterprise information. *International Journal of Production Research*, 43(20):4471–4492, 2005.

[53] M. S. Fox and J. Huang. An ontology for static knowledge provenance. In P. Bernus and M. S. Fox, editors, *Knowledge Sharing in the Integrated Enterprise - Interoperability Strategies for the Enterprise Architect*, pages 203–213, 2005.

[54] T. H. Fran Berman, Geoffrey Fox, editor. *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, Ltd, 2003.

[55] N. Friedman and J. Halpern. Belief revision: A critique. *Journal of Logic, Language, and Information*, 8:401–420, 1999.

[56] F. Fukuyama. *Trust: The Social Virtues and the Creation of Prosperity*. Free Press, 1995.

[57] D. Gambetta. Can we trust trust? In D. Gambetta, editor, *Trust : making and breaking cooperative relations*, pages 213–237. Blackwell, 1988.

[58] D. Gambetta. *Trust : making and breaking cooperative relations*. Blackwell, 1988.

[59] G. Gans, M. Jarke, S. Kethers, and G. Lakemeyer. Modeling the impact of trust and distrust in agent networks. In *AOIS-01 at CAiSE-01*, 2001.

[60] G. Gans, M. Jarke, S. Kethers, G. Lakemeyer, L. Ellrich, C. Funken, and M. Meister. Requirements modeling for organization networks: A (dis-)trust-based approach. In *RE-01*, 2001.

[61] P. Gardenfors. *Belief Revision*. Cambridge University Press, 1992.

[62] M. R. Genesereth and R. E. Fikes. Knowledge interchange format 3.0 reference manual. In *http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps*, 1992.

[63] Y. Gil and V. Ratnakar. Trusting information sources one citizen at a time. In *Proceedings of International Semantic Web Conference*, 2002.

[64] L. Goble. *PThe tipping point : how little things can make a big difference*. Boston : Little, Brown, 2000.

[65] L. Goble. *Philosophical Logic*. Blackwell Publish, 2001.

[66] J. Golbeck, J. Hendler, and B. Parsia. Trust networks on the semantic web. 2002.

[67] A. Gomez-Perez. Tutorial on ontological engineering. In *IJCAI'99*, 1999.

[68] T. W. A. Grandison. *Trust Management for Internet Applications*. Ph.D. Thesis, Imperial College, London, 2003.

[69] E. Gray, J. Seigneur, Y. Chen, and C. Jensen. Trust propagation in small worlds. In *Proceedings of the First International Conference on Trust Management*, 2003.

[70] P. Groth, S. Jiang, S. Miles, S. Munroe, V. Tan, S. Tsasakou, and L. Moreau. An architecture for provenance systems. November 2006.

[71] T. R. Gruber. The role of common ontology in achieving sharable, reusable knowledge base. Morgan Kaufmann, San Mateo, CA, 1991.

[72] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knnowledge Representation*. Kluwer, 1993.

[73] T. R. Gruber. A translation approach to portable ontology specifications. In *KSL-92-71, Knowledge Systems Laboratory, Stanford University*, 1993.

[74] M. Gruninger and M. Fox. Methodology for the design and evaluation of ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-1995*, 1995.

[75] M. Gruninger and C. Menzel. The process specification language (psl) theory and applications. *AI Magazine*, 24(3):63–74, 2003.

[76] N. Guarino. Formal ontology and information systems. In *Proceddings FOIS'98, Italy*, 1998.

[77] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, Amsterdam, 1995.

[78] R. Guha. *Contexts: A Formalization and Some Applications*. Ph.D. Thesis, Standford University, 1995.

[79] R. Guha and R. Kumar. Propagation of trust and distrust. In *WWW2004*, 2004.

[80] R. H. Guttman, A. G. Moukas, and P. Maes. Agent-mediated electronic commerce: a survey. *Knowledge Engineering Review*, 1998.

[81] A. Hajek. Probability, logic, and probability logic. 2001.

[82] J. Y. Halpern. *Reasoning about uncertainty*. MIT Press, 2003.

[83] P. J. Hayes. A catalog of temporal theories. 1995.

[84] D. Heckerman. Probabilistic interpretations for macin's certainty factors. pages 167–196, 1986.

[85] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[86] J. Huang and M. S. Fox. Dynamic knowledge provenance. In *Proceedings of Business Agents and Semantic Web Workshop*, pages 11–20, 2004.

[87] J. Huang and M. S. Fox. Uncertainty in knowledge provenance. In C. Bussler, J. Davies, D. Fensel, and R. Studer, editors, *The Semantic Web: Research and Applications*, Lecture Notes in Computer Science, 3053, pages 372–387, 2004.

[88] J. Huang and M. S. Fox. Trust judgment in knowledge provenance. *dexa*, 00:524–528, 2005.

[89] J. Huang and M. S. Fox. An ontology of trust – formal semantics and transitivity. In *Proceedings of The Eighth International Conference on Electronic Commerce*, pages 259–270. ACM, 2006.

[90] J. Huang and M. S. Fox. Knowledge provenance in financial information. In *EIL Technical Report, University of Toronto*, 2007.

[91] J. Huang and M. S. Fox. Knowledge provenance ontology in a complete logic program. In *EIL Research Report, University of Toronto*, 2007.

[92] J. Huang and M. S. Fox. Knowledge provenance: Web ontology and reasoner. In *EIL Research Report, University of Toronto*, 2007.

[93] J. Huang and M. S. Fox. Uncertain model of knowledge provenance. In *EIL Research Report, University of Toronto*, 2007.

[94] J. Huang and M. S. Fox. Uncertain model of trust propagation in social networks. In *EIL Technical Report, University of Toronto*, 2007.

[95] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, and B. Odgers. Autonomous agents for business process management. *Int. Journal of Applied Artificial Intelligence*, 14, 2000.

[96] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[97] A. Josang. Trust management for e-commerce. 2000.

[98] A. Josang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems*, 9(3):279–311, 2001.

[99] A. Josang, L. Gray, and M. Kinateder. A model for analysing transitive trust. 2005.

[100] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[101] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley & Sons, Inc., 1976.

[102] R. Khare and A. Rifkin. Weaving a web of trust. *World Wide Web Journal*, 2(3):77–112, 1997.

[103] S. C. Kleene. *Introduction to metamathematics*. Princeton, N.J. D. Van Nostrand, 1952.

[104] S. C. Kleene. *Mathematical Logic*. John Wiley & Sons, Inc., 1967.

[105] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[106] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of 32nd ACM Symposium on Theory of Computing*, 2000.

[107] P. R. Kleindorfer, H. C. Kunreuther, and P. Schoemaker. *Decision Sciences: An Integrative Perspective*. Cambridge University Press, 1993.

[108] S. X. Komiak and I. Benbasat. Understanding customer trust in agent-mediated electronic commerce, web-mediated electronic commerce, and traditional commerce. *Information Technology and Mangement*, 5:181–207, 2004.

[109] S. Kripke. Semantic analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1962.

[110] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Rec.*, 31(2):84–93, 2002.

[111] Y. W. Lee, D. M. Strong, B. K. Kahn, and R. Y. Wang. Aimq: a methodology for information quality assessment. *Information and Management*, 40(2):133–146, 2002.

[112] H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. MIT Press, 2001.

[113] J. Lewis and A. Weigert. Trust as a social reality. *Social Forces*, 63(4):967–985, 1985.

[114] J. Li, H. Boley, V. Bhavsar, and J. Mei. Expert finding for ecollaboration using foaf with ruleml rules. In *Montreal Conference on eTechnologies, 2006*, 2006.

[115] J. Lloyd. *Foundations of Logical Programming*. Springer-Verlag, 1984.

[116] J. Lloyd. *Foundations of Logical Programming, 2nd Ed.* Springer-Verlag, 1987.

[117] M. F. Lopez. Overview of methodologies for building ontologies. In *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, 1999.

[118] N. Luhmann. *Trust and Power*. John Wiley & Sons Ltd, 1979.

[119] D. MacKay. *Information Theory, Inference, and Learning Algorithm*. Cambridge University Press, 2003.

[120] P. Maes, R. Guttman, and A. Moukas. Agents that buy and sell: Transforming commerce as we know it. *Communications of the ACM*, 1999.

[121] G. Malinowski. Many-valued logics. 2001.

[122] S. P. Marsh. *Formalising Trust as a Computational Concept.* Ph.D. Thesis, University of Stirling, 1994.

[123] S. P. Marsh and M. R. Debben. Trust, untrust, distrust and mistrust - an exploration of the dark(er) side. In *Proceedings of iTrust2005, LNCS 3477*, pages 17–33, 2005.

[124] R. Mayer, J. Davis, and F. Schoorman. An integrative model of organizational trust. *Academic of Management Review*, 20(3):709–734, 1995.

[125] A. Mayorkas. *http://web.archive.org/web/20021209030248/ http://www.usdoj.gov/usao/cac/pr/pr2000/003.htm*, 2000.

[126] J. McCarthy. Notes on formalizing context. In *Proceedings of IJCAI1993*, 1993.

[127] D. McGuinness and P. da Silva. Infrastructure for web explations. In *Proceedings of 2nd International Semantic Web Conference*, pages 113–129, 2003.

[128] D. McKnight and C. N.L. Conceptualizing trust: A typology and e-commerce customer relationships model. In *Proceedings of 34th Hawaii Int. Conf. on System Sciences*, 2001.

[129] J.-J. C. Meyer. Epistemic logic. 2001.

[130] J.-J. C. Meyer and W. der Hoek. *Epistemic Logic for AI and Computer Science.* Cambridge University Press, 1995.

[131] S. Milgram. The small world problem. *Psychology Today*, 61(1), 1967.

[132] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja1, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. 2002.

[133] N. Minsky. Regularity-based trust in cyberspace. In *PProceedings of 1st International Conference on Trust Management, Lecture Notes in Computer Science, 2692*, pages 17–32. Springer, 2003.

[134] MIT-SDSI-project. A simple distributed security infrastructure.

[135] L. Moreau. Provenance: an open approach to experiment validation in e-science. August 2006.

[136] L. Moreau and J. Ibbotson. The eu provenance project: Enabling and supporting provenance in grids for complex problems (final report). December 2006.

[137] L. Mui and A. Halberstadt. A computational model of trust and reputation. In *Proceedings of the 35th Hawaii International Conference on System Sciences*, 2002.

[138] F. Naumann and C. Rolker. A computational model of trust and reputation. In *Assessment Methods for Information Quality Criteria*, October 2000.

[139] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12, 1991.

[140] N. Noy and D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. In *Knowledge Systems Laboratory, Stanford University*, 1993.

[141] K. Oliver. Evaluating the quality of internet information. 1997.

[142] Oxford. *Oxford Dictionary of Current English, 4th Edition.* Oxford University Press, 2006.

[143] M. Paolucci and K. Sycara. Autonomous semantic web services. *IEEE Internet Computing*, 7(5):34–41, 2003.

[144] J. Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference.* Morgan Kaufmann Publishers, 1988.

[145] A. G. Perez and V. Benjamins. Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. In *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, pages 25–32. IOS Press, Amsterdam, 1995.

[146] PGP. Pgp home page.

[147] J. Pinto. *Temporal Reasoning in the Situation Calculus*. Ph.D. Thesis, University of Toronto, 1994.

[148] L. Pipino, Y. Lee, and R. Wang. Data quality assessment. *Communications of ACM*, 45(4):211–218, 2002.

[149] S. Ramchurn, H. Dong, and N.R.Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.

[150] R. Reiter. *Knowledge In Action*. The MIT Press, 2001.

[151] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[152] R. R.Guha and R.Fikes. Contexts for the semantic web. 2005.

[153] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceedings of International Semantic Web Conference*, pages 351–368, 2003.

[154] S. Y. Rieh and N. J. Belkin. Understanding judgment of information quality and cognitive authority in the www. In *Proceedings of the 61st Annual Meeting of the American Society for Information Science*, pages 279–289, 1998.

[155] J. Rotter. A new scale for the measurement of interpersonal trust. *J. Personality*, 35:651–665, 1967.

[156] D. M. Roussea, S. B. Sitkin, R. S. Burt, and C. Camerer. Not so different after all: A cross-discipline view of trust. *Academic of Management Review*, 23(3):393–404, 1998.

[157] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[158] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, 2nd Edition*. Prentice Hall, 2003.

[159] L. Schamber. *Users' Criteria for Evaluation in Multimedia Information Seeking and Use Situations*. Ph.D. Thesis, Syracuse University, 1991.

[160] R. Scherl and H. Levesque. The frame problem and knowledge-producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 689–695, 1993.

[161] V. Shankar, F. Sultan, and G. Urban. Online trust and e-business strategy: Concepts, implications, and future directions. 2002.

[162] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[163] Y. Shaoham. Temporal logics in ai: Semantical and ontological considerations. *Artificial Intelligence*, 33:89–104, 1987.

[164] Y. Shaoham. *Reasoning About Change*. The MIT Press, 1988.

[165] S. Shapiro, Y. Lespérance, and H. J. Levesque. Specifying communicative multi-agent systems with congolog. In *In Working Notes of the AAAI Fall 1997 Symposium on Communicative Action in Humans and Machines*, pages 72–82, Cambridge, MA, Novemeber 1997. AAAI Press.

[166] S. Shapiro, M. Pagnucco, Y. Lespénce, and H. J. Levesque. Iterated belief change in the situation calculus. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference, (KR2000)*, San Francisco, CA, 2000. Morgan Kaufmann.

[167] E. Simon, P. Madsen, and C. Adams. An introduction to xml digital signatures. 2001.

[168] E. Simon, P. Madsen, and C. Adams. Building your appropriate certificate-based trust mechanism for secure communications. 2002.

[169] H. A. Simon. *Models of Bounded Rationality*, volume 3. The MIT Press, 1997.

[170] H. A. Simon. *Models of Bounded Rationality*, volume 2. The MIT Press, 1997.

[171] A. Smith. Testing the surf: Criteria for evaluating internet information resources. *The Public-Access Computer Systems Review*, 8(3), 1997.

[172] M. Stanojevic, S. Vranes, and D. Velasevic. Using truth maintenance systems: A tutorial. *IEEE Intelligent Systems*, 9(6):46–56, 1994.

[173] W. trust group. Web services trust language. February 2005.

[174] UC-Berkeley-Library. Critical evaluation of resources. In *http://www.lib.berkeley.edu/TeachingLib/Guides/Evaluation.html*, 2002.

[175] M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11, 1996.

[176] M. Uschold and R. Jasper. A framework for understanding and classifying ontology applications. In *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, 1999.

[177] W3C. Resource description framework (rdf) model and syntax specification. February 1999.

[178] W3C. Xml-signature syntax and processing. February 2002.

[179] W3C. Owl web ontology lauguange guide. February 2004.

[180] W3C. Owl web ontology lauguange reference. February 2004.

[181] W3C. Rdf vocabulary description language 1.0: Rdf schema. February 2004.

[182] N. Walsh. A technical introduction to xml. October 1998.

[183] R. Wang, H. Kon, and S. Madnick. Data quality requirements analysis and modeling. In *Proceedings of the Ninth International Conference of Data Engineering*, pages 670–677, April 1993.

[184] R. Wang, M. Ziad, and Y. Lee. *Data Quality*. Kluwer Academic Publishers, 2001.

[185] Wikipedia. Social network. 2005.

[186] P. Wilson. *Second-Hand Knowledge*. Greenwoods Press, 1983.

[187] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons Ltd., 1997.

[188] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10:115–152.

[189] XBRL-International. Extensible business reporting language (xbrl) 2.1. 2005.

[190] XML.Query.WG. Xquery 1.0: An xml query language. 2006.

[191] R. Yahalom, B. Klein, and T. Beth. Trust relationships in secure systems - a distributed authentication perspective. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, 1993.

[192] B. Yu and M. Singh. A social mechanism of reputation management in electronic communities. In *Proceedings of Fourth International Workshop on Cooperative Information Agents*, pages 154–165, 2000.

[193] E. Yu and L. Liu. Modelling trust for system design using the i* strategic actors framework. In R. Falcone, M. Singh, and Y. Tan, editors, *Trust in Cyber-Societies - Integrating the Human and Artificial Perspectives, LNAI-2246*, pages 175–194. Springer Verlag, 2001.

[194] J. Zhang and R. Cohen. Trusting advice from other buyers in e-marketplaces: the problem of unfair ratings. In *Proceedings of The Eighth International Conference on Electronic Commerce*, pages 225–234. ACM, 2006.

[195] P. Zimmermann. *The official PGP user's guide*. MIT Press, 1995.

[196] L. Zucker. Production of trust: Institutional sources of economic structure, 1840-1920. *Research in Organizational Behavior*, 8:53–111, 1986.