

Universität Stuttgart

Sonderforschungsbereich 627

Umgebungsmodelle für
mobile kontextbezogene Systeme

SFB 627 Bericht Nr. 2010/02

Reference Model for the Quality of Context Information

Datum: 11. Februar 2010

Autor(en): Susanne Becker, André Blessing,
Frank Dürr, Lars Geiger,
Matthias Großmann, Andreas Gutscher,
Kai Häussermann, Jessica Heesen,
Uwe-Philipp Käppeler, Ralph Lange,
Michael Peter, Oliver Siemoneit,
Harald Weinschrott, Oliver Zweigle,
Paul Levi, Kurt Rothermel

CR Klassifikation: G.1.2, G.3, H.3.1, H.3.3, I.2.6

(c) 2010 Susanne Becker, André Blessing,
Frank Dürr, Lars Geiger,
Matthias Großmann, Andreas Gutscher,
Kai Häussermann, Jessica Heesen,
Uwe-Philipp Käppeler, Ralph Lange,
Michael Peter, Oliver Siemoneit,
Harald Weinschrott, Oliver Zweigle,
Paul Levi, Kurt Rothermel

Center of Excellence 627
Spatial World Models for
Mobile Context-Aware Applications

Sprecher des SFB:
Prof. Dr. Kurt Rothermel
Institut für Parallele und Verteilte Systeme
Universitätsstraße 38
70569 Stuttgart
Deutschland

NEXUS

www.nexus.uni-stuttgart.de

Abstract

Context-aware applications require context information for their operation. However, context information is inherently associated with uncertainties, which need to be taken into account when processing such information. This *Quality of Context Information* (QoC) comprises numerous aspects, such as the uncertainty of sensed data, transmission and update protocols, consistency between data from several providers, or the trust placed into the information from individual providers.

Because of the technical and economical challenges, it is our vision that a federation integrates many context models from different providers into a global context model. This model is then shared by a number of applications. Such a federated context management poses additional challenges with regard to QoC. Therefore, a universal QoC model is needed, which covers and integrates the different aspects of QoC on all abstraction levels.

To enable a federated context management system such as Nexus to incorporate QoC, we present a *QoC reference model* in this document. Specifically, it consists of five main parts:

1. *Abstract framework for quality aspects:* This framework distinguishes three abstraction levels of context information – sensor information, observable context, and high-level context –, as well as three fundamental quality aspects – degradation, consistency, and trust.
2. *Degradation model:* Sensed data is typically inaccurate due to physical limitations of sensors or the employed update protocols.
3. *Consistency model:* Context information – even with degradation information – may be inconsistent to the physical world or between different providers.
4. *Trust model:* With arbitrary context providers, a comprehensive approach for assessing the reliability and trustworthiness of each provider is necessary.
5. *Integrated QoC-aware processing model:* We propose a universal QoC-aware processing model for queries on context information. It incorporates the specific models for degradation, consistency, and trust and their dependencies.

Contents

1	Introduction	11
2	Overview to Reference Model	15
2.1	Requirements and Challenges	15
2.2	Abstraction Layers of Context Information	16
2.3	Quality Aspects of Context Information	17
2.4	Abstract Framework for Quality of Context	18
3	Degradation Model	21
3.1	Universal Model for the Quality of Sensor Data	22
3.1.1	Modeling Quality of Sensor Data	22
3.1.2	Reliable updates for information in environmental models using sensors	25
3.2	Generic Uncertainty Model for Position Information	30
3.2.1	Introduction	30
3.2.2	Survey of Uncertainty Models	32
3.2.3	Mathematical Generalization for Time-dependent Point Data	33
3.2.4	Uncertainty-aware Query Interface for Position Information	35
3.2.5	Related Work	39
3.2.6	Summary	40
3.3	Uncertainty Model for 3D Geodata	40
3.4	Uncertainty-Aware Situation Detection with Bayesian Networks	43
3.4.1	Introduction	43
3.4.2	Situation Template Model	43
3.4.3	Parameter-Adaption	47
3.4.4	Summary	51
3.5	Degradation of high-level context derived from sensor data	52
4	Consistency Model	57
4.1	Inconsistency on the Context Information Layer	57
4.1.1	Discrete Domain	58
4.1.2	Continuous Domain with Pdf	58
4.1.3	Continuous Domain without Pdf	59
4.1.4	3D-Domain	60
4.2	Situation Recognition	62
5	Trust Model	65
5.1	Introduction	65

Contents

5.2	Related Work and Fundamentals	66
5.2.1	Trust, Trustworthiness and Reputation	66
5.2.2	Modeling Trust	69
5.2.3	Classification of Reputation Systems	71
5.2.4	Reasoning with Trust Relations	72
5.2.5	Computation of Reputation Values	73
5.3	Model of Trust and Authenticity Statements	78
5.4	Trust Values	79
5.4.1	Representation of Discrete and Continuous Trust Values	80
5.4.2	Deterministic Operators	81
5.5	Inference Rules	85
5.5.1	Transitive Trust Inference Rule	85
5.5.2	Trust in Entities, Keys and Descriptions	85
5.5.3	Local Authenticity Inference Rule	87
5.5.4	Authenticity Inference with Authenticity Confirmation	87
5.5.5	Uniqueness Conditions	87
5.6	Trust Value Computation	87
5.6.1	Probabilistic Model for the Trust Value Computation	88
5.6.2	Approximation and Exact Computation Algorithms	89
5.7	Evaluation of the Proposed Trust Model	95
5.7.1	Computation Time of the Proposed Computation Algorithms	95
5.7.2	Comparison with Other Trust Models	98
5.7.3	Trade-offs and Limitations	99
5.8	Reliability of Context Information	99
5.9	Summary	101
6	Reference Model	103
6.1	Interdependencies between Uncertainty, Consistency, and Trust	103
6.2	Integrated Quality-Aware Processing Model	104
6.2.1	The Nexus System	105
6.2.2	Three Aspects of Quality of Data	106
6.2.3	Query Processing	107
6.2.4	Processing Model	109
6.2.5	Revisiting the Example Scenario	112
7	Conclusions	115
	Bibliography	117

List of Figures

2.1	3×3 framework for quality of context.	19
3.1	Singular behavior: Absolute deviations in a simulation of 1000 negotiations	28
3.2	1000 consecutive simulated negotiations with 5 agents. The deviation of one sensor is artificially set to 0 in negotiation 400	29
3.3	1000 consecutive simulated negotiations with 5 agents. Artificially noise is added to the measurements of one sensor from negotiation 400 onward.	30
3.4	Taxonomy of major classes of the existing uncertainty models.	34
3.5	Mathematical generality of the major uncertainty models.	34
3.6	Distance query evaluation.	36
3.7	2D normal distribution.	37
3.8	Range query evaluation.	37
3.9	Uncertainty region of a plane (from [Haa96]).	41
3.10	Exemplary Nexus applications using uncertainty information of a 3D building model. <i>Left:</i> Inside query based on a global uncertainty value of the building. <i>Middle:</i> Range query based on local uncertainty measures. <i>Right:</i> Navigation task based on local error descriptions.	42
3.11	Situation Template: SAT-Structure.	47
3.12	Screenshot of Template Designer.	48
4.1	Temperature specified by probability density functions	59
4.2	Uncertain areas and corresponding probabilities	59
4.3	Clockwise: Inconsistencies of OpenStreetMap model, generalized model and city model from airborne data collection in comparison to the city model from terrestrial data collection (upper left)	62
4.4	Structure of a meta-template	63
5.1	Exemplification of trust types and categories (trust context “landing a plane”)	68
5.2	Reputation System	68
5.3	Different possibilities to represent trust values	70
5.4	Classification of reputation systems	71
5.5	Operator-based reputation computation	74
5.6	Deterministic conjunction, disjunction and negation operators (Baldwin, Jøsang)	74
5.7	Recommendation operator (Jøsang)	75

List of Figures

5.8	Jøsang's (top), Dempster-Shafer's (middle) and Yager's (bottom) consensus operators	75
5.9	Probability theoretical reputation computation	77
5.10	Our deterministic conjunction, disjunction and negation operators	82
5.11	Consensus, recommendation and authentication operator truth tables	83
5.12	Example for application of the transitive trust inference rule	86
5.13	Scenario and possible worlds table of the Trust Example 1	90
5.14	Scenario of the Trust Example 2	92
5.15	Scenario of the Trust Example 3	94
5.16	Scenario of the simple chain example	96
5.17	Computation time in the simple chain example	96
5.18	Scenario of the full mesh example	97
5.19	Computation time in the full mesh example	97
6.1	Interdependencies between Degradation, Consistency, and Trust	104
6.2	Architecture of the Nexus system [LCG ⁺ 09a]	105
6.3	Extending the possible worlds model to support uncertainty	107
6.4	Example scenario: PDFs of the positions of o_1 and o_2	108
6.5	Measuring the quality of the query result	109
6.6	Processing model (data providers, trust and reliability)	110
6.7	Processing model (data fusion and query processing)	110
6.8	Processing the query ($v(o_1^1.p) = v(o_2^1.p) = v(o_2^2.p) = 1$)	113
6.9	Processing the query ($v(o_1^1.p) = v(o_2^1.p) = 0.5, v(o_2^2.p) = 1$)	113

List of Tables

3.1	Example of an extension table T_{cv_i} with fictitious context data. . . .	51
4.1	Example values	58
4.2	Inconsistent position information and resulting consistency values . .	60
5.1	Trust statements (relations and certificates)	78
5.2	Authenticity statements (relations and certificates)	79
5.3	Discrete trust values	80
5.4	Preparation step for the groups in the Trust Example 2	92
5.5	Trust value computation in the Trust Example 2	92
5.6	Propositions and applied inference rules in the Trust Example 3 . . .	94

1 Introduction

Advances in sensor technology, wireless communication, and mobile devices allow for applications that take into account the context of entities of the physical world such as current locations of users, climatic conditions, and user activities. These *context-aware* applications adapt to changes in the physical world and select and present information depending on their context. The required *context information* can be acquired from associated sensors (e.g., embedded GPS-receiver of smartphones) as well as retrieved from digital *context models* of clippings of the physical world provided by various sources. Context models often combine static spatial data such as maps and building plans with dynamic information from numerous sensors and can be used for a variety of applications.

Context information is inherently associated with uncertainties which must be taken into account by context management as well as context-aware applications. We say context management and applications have to factor in the *Quality of Context Information* (QoC).

QoC comprises many different aspects such as the uncertainty of sensed data values, the timeliness of transmission protocols from remote sensors, the trustability of providers of context models, the reliability of sensors, the consistency between providers of context information on the same phenomenon of the real world, and many more. Therefore ontologies, models, and metrics are required to specify the different aspects of QoC for context management and applications. In addition, different levels of abstraction of context information have to be taken into account, including raw sensor data, observable context, and high-level context deduced by situation recognition and reasoning techniques.

Consider, for example, a context-aware call manager for mobile phones. To decide whether and how to notify the user on an incoming call, it has to consider the position and activity of the user, the time of day, the user's calendar, the relation of the caller to the user, etc. The decision process involves a number of uncertainties such as the accuracy of the sensed position, the certainty of activity recognition, and the (typical) actuality of the user's calendar. This requires metrics for the uncertainty of position sensing and activity recognition methods, depending on the accuracy of the underlying input, as well as models how to specify and acquire the actuality of the calendar. To be able to account for wants of the user regarding uncertain situations (e.g., if the call manager is not able to distinguish whether he gives a talk or just attends a meeting), the decision process has to consider the QoC in an integrated fashion, from the raw data to the final decision.

For economic and technical reasons, it is highly desirable for context information to be shared by context-aware applications. In the Nexus project we envision—in analogy to the WWW—a World Wide Space, which provides the conceptual and

1 Introduction

technological framework for integrating and sharing context models. It enables arbitrary commercial and non-commercial providers to make context models available for a wide range of context-aware applications and services. The collection of context models is federated and leads to a large-scale context model, offering a global view on the available context information. The federation allows for complex spatial queries, including continuous, stream-based processing.

Federated context-management by a framework such as Nexus poses additional requirements to the models and metrics for QoC. A universal model for QoC is needed which covers and integrates the different aspects of QoC at all levels of abstractions—not only for a specific application but comprehensively for a wide range of applications. This particularly also includes methods and calculi for assessing the quality of context information between different abstraction layers as well as dependencies between different aspects of QoC.

Consider, for example, a 3D building model combining a static building plan with dynamic information on the state of the doors and windows, obstacles (e.g., recognized using the cameras of smartphones), room occupancies, and so on. This model may be used for a number of purposes, including navigation of blind persons and energy-efficient building automation. However, both applications pose very different requirements to the QoC of the model—e.g., consider the occupancy of a glazed meeting room: A blind person may want to be warned if there is a slight probability that the meeting room is occupied, whereas the heating temperature should be only increased if the room is likely occupied. Therefore, boolean statements on the room occupancies are not sufficient, but the building model has to provide the reliability of this information as well. Since the occupancies again have to be deduced from sensors, wireless network connections, and calendars, an integrated and comprehensive approach how to cope with QoC from the raw sensor data to the reliability of occupancy statements is needed, which fits for both applications.

In this report, we present a *reference model* for QoC which meets the requirements for federated context-management—in particular for Nexus. The reference model consists of five parts:

1. *Abstract framework for quality aspects:* We propose an abstract framework for QoC which distinguishes between three fundamental abstraction levels of context information, namely sensor information, observable context, and high-level context, as well as three fundamental quality aspects, namely degradation, consistency, and trust.
2. *Degradation model:* We propose an integrated approach how to cope with physical limitations of sensors and the corresponding sensing inaccuracies, imprecision, etc.—from the raw sensor data, via observable context to high-level context. Moreover, for applications, we present QoC-aware interfaces for querying context information.
3. *Consistency model:* Context information with assessed degradation may be inconsistent when different context data providers disagree about the value of a

context data item. Such disagreements may be caused by incorrect degradation information, typing errors or tampering. We present a metrics for measuring consistency considering the degradation model.

4. *Trust model:* With an open context-management platform such as Nexus, trustworthiness of context providers is an important issue. For this reason we propose a comprehensive approach for assessing the reliability and trustworthiness of each provider.
5. *Integrated QoC-aware processing model:* We propose a universal QoC-aware processing model for queries for context information of arbitrary applications. This model incorporates the specific models for degradation, consistency, and trust and the dependencies between these aspects.

The remainder of this report is structured as follows: In Chapter 2 we render the requirements and challenges to the reference model more precisely and discuss the abstract framework for QoC, before we present the degradation model in Chapter 3. In Chapter 4 we present the consistency model followed by the trust model in Chapter 5. In Chapter 6 we present the integrated QoC-aware processing model. Finally, the report is concluded in Chapter 7 with a summary and an outlook.

2 Overview to Reference Model

In this chapter, we provide an overview of the developed reference model. We begin by deriving the requirements for this model as well as the challenges it poses. Then, we introduce the different abstraction layers for context information, which we use in our reference model. The different aspects of Quality of Context is the topic of the next Section, before finally putting all of these parts together to form our abstract framework for Quality of Context.

2.1 Requirements and Challenges

As introduced and exemplified in Chapter 1, the reference model has to allow for integrated quality-aware processing of context information in federated context management. This particularly requires a generic approach which is appropriate for different types applications and their needs. We distinguish three fundamental requirements to the reference model:

1. *Comprehensiveness*: The reference model has to account for the various quality aspects depending on different types of context information and their abstraction levels. In detail, it has to cover all basic quality aspects such as accuracy, precision, timeliness, consistency, and trustworthiness for all types of context information including time-dependent numerical data, complex spatial information, and textual context information at all abstraction levels from raw sensor data to high-level logical statements deduced by situation recognition and reasoning techniques. This also requires to define QoC-aware interfaces for applications at different abstraction levels since applications may process simple, sensed context information just as high-level context information at the same time.
2. *Universality*: Federated context management aims at supporting a variety of context-aware applications and services with diverse needs regarding QoC: Some applications pose strict requirements to the quality of context information and cannot cope with less. Others are able to cope with any degree of quality—e.g., they may just visualize the uncertainty to the user correspondingly. Other, legacy applications, simply ignore QoC and thus require corresponding methods to retrieve the most probable state of the physical world.
3. *Processability*: The reference model has to allow for integrated QoC-aware processing between different abstraction levels of context information. It has to provide the foundation for mapping quality aspects from lower levels (e.g.,

of sensor data) to higher levels (e.g., recognized situations). Similarly it has to allow for mapping the requirements of applications from higher levels to lower levels and to adapt the processing within the federation accordingly.

These requirements lead to a number of challenging research questions: How to define appropriate abstraction levels for context information that apply to all types of context? How to classify the fundamental aspects of quality? How to model and measure each quality aspect at each abstraction level comprehensively? How do the models and metrics depend on the type of context information? How to map between these models and metrics at different abstraction levels—e.g., from the Dilution of Precision (DOP) values provided by GPS to a generic uncertainty measure for position data? How to map between these models and metrics when processing different types of context information together—e.g., to answer whether an uncertain position is inside an uncertain spatial range or to derive a probability value for a situation recognized by a Bayesian network?

We propose an answer to the first two questions on the following two sections and present the resulting abstract framework for QoC in Section 2.4. The other questions are tackled in the Chapters 3 to 6.

2.2 Abstraction Layers of Context Information

Context-aware systems are widely considered as layered systems. The survey [BDR07] introduces common architecture principles of context-aware systems and derives a layered conceptual design framework that allows to classify the different elements of context-aware architectures. In essence, [BDR07] identifies five abstraction layers. The lowest layer consists of sensors, i.e., the sources of context data. The next layer abstracts from raw sensor data by providing context information. In the third layer, reasoning on and interpretation of context information is performed. The fourth layer deals with managing the context models and providing them synchronously or asynchronously to the applications, which compose the fifth layer. Similar to this generic layered architecture of context-aware systems, works such as [HIMB05, FC04, HSP⁺03, KMK⁺03, WDC⁺04] also consider layered architectures for their systems.

In accordance with the principle of a layered architecture for context-aware systems, we identified the three layers of such systems that are relevant for quality of context (QoC) considerations as layers one to three. First, the sensor layer deals with raw sensor data that is acquired. Then, the second layer produces context information from this raw data by associating it to context models. This layer relates the sensor data to static context information to provide a dynamic object-oriented context model of the physical world. For instance, it attaches the position data of a GPS receiver which is embedded into a car to the corresponding car object in the context model. Finally, the third layer derives higher level context by reasoning and situation recognition techniques from the context information provided by the second layer. These three types of information are handled by context-aware systems

and, therefore, in the following, we consider this simplistic, but universal, three layer model.

One example for data on these layers is position information. Positioning systems as GPS or RFID can serve as source for position data and report specific raw sensor data such as NMEA data sentences [Nat02] with longitude and latitude values in case of GPS receivers. The context layer associates the position data to the corresponding moving object such as cars which makes up the context information about this object. It also hides data specifics by representing position information in a generic format. In the third layer, a meeting might be detected by inferring from position information of people in a room and their calendars.

2.3 Quality Aspects of Context Information

Context information mostly is acquired by sensors and therefore subject to physical limitations and measurement errors of sensing hardware. Depending on the type of context information such as position, temperature, open/closed states, etc. these limitations and errors lead to inaccuracies, imprecisions, delays and other kinds of degradation. The context information may be further degraded due to networking and processing delays and conversions between different schemas. Therefore, we subsume all these quality aspects under the term *degradation* in the following. Note that the degradation by conversion and processing generally applies to all types of context information, not only to sensor data. Mathematical models for the degradation of context information typically base on probability distributions, tolerance regions, or statistical measures.

Context information with defined degradation may nevertheless be inconsistent to the physical state of the world—e.g., due to failures of sensors, human errors, or even tampering. Such inconsistencies can be only detected by redundant observations of the same phenomenon. A federated context management platform particularly allows to detect such inconsistencies by means of contradictory context information of different (independent) providers. Hence, inconsistencies between the context models of providers may be used to detect inconsistencies to the physical world. *Consistency* therefore is another integral aspect of quality of context information in an open context management platform such as Nexus. Defining appropriate concepts to assess the consistency between providers is a difficult task given the degradation of context information. In general, the consistency can be only evaluated by probabilistic measures.

The possibility of tampering of data or sensors points to the third fundamental quality aspect namely *trust*. An open context platform comprises a large number of different actors including users, service providers, application vendors, network providers, sensor manufacturers, and providers of context information. Users have different requirements regarding the trustworthiness of providers, etc. depending on personal experiences and attitudes as well as the use of context information. Therefore, an open context platform must inherently feature approaches to assess the trustworthiness of providers and to select trustworthy providers correspondingly.

2.4 Abstract Framework for Quality of Context

To summarize the previous two sections, an abstract framework for providing context-information in a quality-aware manner needs to comprise the three quality aspects, *degradation*, *consistency* and *trust*. Each of the three facets needs to be modelled along the layers of the system: from the low level sensor data via the context information layer up to the higher level context in such a system. The processing steps can also influence the quality of the data, so the processing steps need to take this into account as well. This leads us to an abstract framework as shown by the 3×3 matrix in Figure 2.1.

The different quality aspects are — at first glance — rather independent: it is possible to discuss the degradation of information without any knowledge about the trust in the source or the consistency of the information and vice versa. Therefore, in Chapters 3 through 5, we first show how the degradation of numerical values – using position data as an example – can be modelled for different sensors and their specific uncertainty models. We also show how applications can access this quality information in a consistent manner via a defined uncertainty-aware query interface. After that, we present an approach to handle multiple pieces of information with regard to the same real world phenomenon and the inconsistencies that arise from it. And finally, we show how a trust model allows us to judge the correctness of data according to the trust we place in sources or indirectly via the trust in third parties making statements about the correctness.

These building blocks provide the vertical processing for data quality in a context-aware system such as Nexus. Chapter 6 shows our model for a quality-aware, horizontally integrated processing of queries for context information.

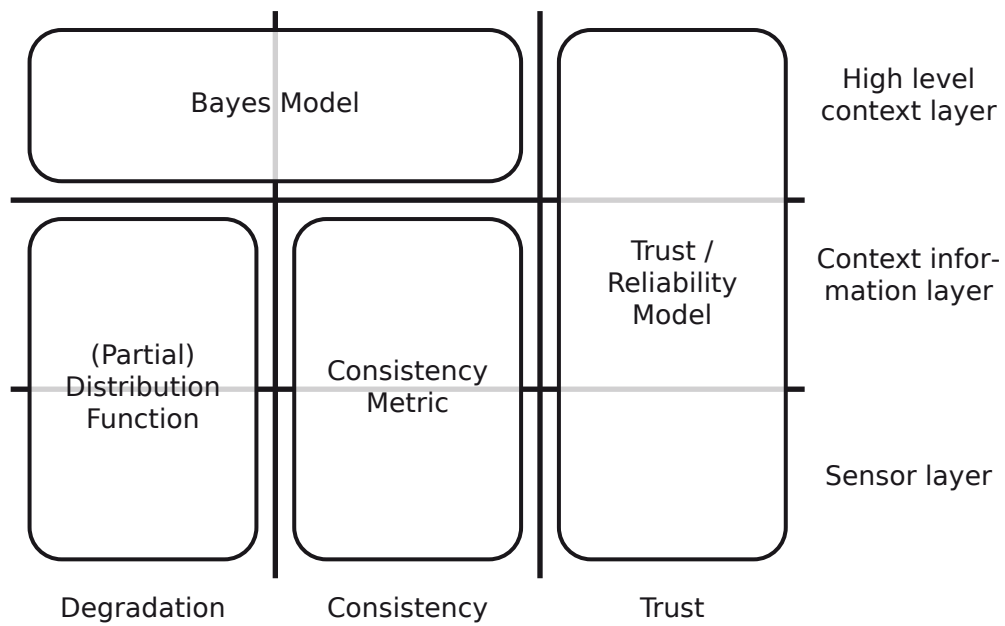


Figure 2.1: 3×3 framework for quality of context.

3 Degradation Model

In contrast to other parts of the quality reference model, degradation in general describes quality of information without any comparison to other information sets and without any subjective rating from users [DKN⁺06]. A quantification for degradation is derived from its source when obtaining or updating data in an environmental model. Basically quality of degradation is deduced from physical limitations of sensors or from an error analysis performed by data fusion algorithms which were used to generate the accordant information. This chapter describes summarized

- a modeling of degradation of numeric context information,
- handling of uncertainty of numeric context information without given semantics,
- handling of uncertainty of position information,
- handling of uncertainty of complex information like building models,
- handling uncertainty of methods for situation detection
- and methods for transfer degradation of context to highlevel context

Defining different aspects of degradation we distinguish between different types of information. First of all degradation of sensor data and context information is described. In both cases it is possible to derive the degradation from attributes of data sources like sensors and their physical constraints. But in an open system like the nexus platform it is necessary to give a first definition for degradation without required assumptions about semantics of the information or required knowledge of applications and their use cases which have access to the rated information. Only statistical methods can be used to define and handle degradation of information when the semantics are no predefined. Modeling of quality of sensor data and a normalized weighted arithmetic mean algorithm which can be used for updates and extension of an environmental model using scalar sensor data is described in section 3.1.

Information in the context layer can be combined with semantics like position information. Therefore it is possible to apply specialized methods for the handling of degradation concerning the semantics of this numeric information. Assumptions concerning values and their history can be used to rate their degradation based on special models like the possible movement of pedestrians, cars, trains or other vehicles. A quality model for position information based on different kinds of distribution functions is described in section 3.2.

3 Degradation Model

Degradation of more complex information consisting of several values can also be handled using predefined assumptions and conditions that must be applicable on linked datasets. We define degradation models for 3D geodata in section 3.3. These models can be used for example to describe and compare 3D models of buildings in different levels of detail.

Concerning high-level context information the degradation in the process of a situation detection is described in section 3.4. This defines degradation for information that can not be observed directly but is derived from context information using specialized knowledge about situations. Finally a method for transferring degradation from context information to high-level context is explained in section 3.5.

3.1 Universal Model for the Quality of Sensor Data

This chapter describes the handling of quality aspects of sensor data. First the model for degradation is described, which is used in the nexus platform by services and applications to provide and process information and its degradation. Afterwards methods are described for extending and updating environmental models. The last section describes procedures to derive higher-level context from sensor data together with a corresponding quality rating.

3.1.1 Modeling Quality of Sensor Data

In this section we describe syntax and semantics of meta data concerning all types of degradation of sensor data. The complete Extended Attribute Schema is listed in [Köp08]. The meta data is attached optionally to single attributes or nexus objects of the environmental model [HKN⁺05].

Different levels of detail for modeling degradation

Modeling of degradation needs different levels of detail to allow for different applications to handle degradation and inaccurate context information in different ways. Therefore one important aspect of modeling degradation is the optional integration of single attributes concerning different details of degradation and attributes that accumulate the information about degradation to a simple and short overall dimension of degradation. This is mandatory in an open system like the Nexus Platform because it is impossible to force all software designers to completely interpret all aspects of degradation in every application as well as it is impossible to force all providers to offer meaningful meta data about degradation for all information in the environmental model.

Applications and services can be classified in three different types in terms of handling the quality of information. First of all there are systems that ignore the quality of information. Other systems get by with a total value for quality that describes the overall degradation. The third type of applications and services

depends on detailed descriptions of several kinds of degradation.

The first type of services and applications refrains from handling any quality aspects of information. Services do not provide quality information and applications just ignore it when processing any context information. This procedure complies with the behaviour of most applications and services down to the present day. For special use cases a desired degree of quality is predetermined assumed implicitly. Data sources and sensors are selected accordingly. No drawback follows from this approach as long as sensors are used for a single purpose or a limited and known amount of applications.

The second type of services and applications processes short total values for quality. One value describing a maximum relative or absolute error enables the processing and comparison of information from different sources. This is mandatory in an open system like the nexus platform without any predetermined assignment of sensors to applications.

Applications can imply different requirements for several aspects of degradation. This can be necessary when fusing data from different sensors. This also enables applications to elect the important aspects of degradation according to an intention. For example some applications might attach importance to up-to-dateness to be able to react rapidly on changes in the environment whilst other applications prefer accurate sources of information for example to initiate any security relevant task.

To satisfy the different requirements concerning the description of quality the corresponding schema in the nexus platform is split into a part consisting of several attributes describing different sorts of degradation and a part consisting of summarizing attributes.

Attributes of meta data concerning a total value of degradation

The following attributes of information can be derived from a sensors manufacturer's data in the majority of cases.

uncertaintyAbsolute

This attribute describes an absolute error. A single floating point value is sufficient for describing the error. The unit has to match to the corresponding attribute, to which the meta data refers, implicitly. The attribute `<uncertaintyAbsolute>` relates to a maximum of aberration.

uncertaintyRelative

This attribute describes an error, whose maximum aberration is depending on the currently measured value itself. The relative error e_r is a function of the measured value x_m and the true value x described by equation 3.1.

3 Degradation Model

$$e_r = \frac{|x_m - x|}{x_m} * 100\% \quad (3.1)$$

uncertaintyReduced

Another possibility to give a description of quality is a reduced error. Here the aberration from the true value is not related to the measured value but described as ratio of the measurement range. The reduced error e_z depending on the lower and upper boundary of the domain x_{min} and x_{max} is described by:

$$e_z = \frac{|x_m - x|}{x_{max} - x_{min}} * 100\% \quad (3.2)$$

uncertaintyStandard

The declaration of a standard deviation σ enables the description of a sensors's Gaussian distribution. It can be determined empirically by n reference measurements as described in equation 3.3.

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{mi} - x_i)^2} \quad (3.3)$$

Degradation of scalar sensor data

Meta data about degradation can be attached to objects or attributes. When corresponding to a value of an attribute, the information about degradation can be derived from the physical attributes of the sensor, that measured that particular value. The meta data can as well be derived from error analysis when several measurements are combined to one value as it is for example when a gps sensor calculates a position from several distance measurements. In addition to the total values for quality described in the previous section there are the following optional elements of the meta data concerning degradation, which describe particular types of degradation in detail. The element **<resolution>** consists of a **<range>** together with an **<lowerBoundary>** and **<upperBoundary>** which gives the domain of a sensors measurements. Then there is a **<quantification>** element that describes the resolution of a sensor and a **<discrimination>** element, describing the minimum change of an observed phenomenon, that leads to a change in the measured value. The element **<processing>** represents influences on the quality produced by signal processing. The element itself consists of **<digitalization>**, **<amplification>** and **<linearization>**. All sorts of temporal aspects are described by the meta data element **<temporal>**. Here the reason for a degradation is described as **<drift>**, **<hysteresis>**, **<responsetime>**, **<samplingrate>** and a **<precision>** which describes the repeat accuracy. The last element is **<crossSensitivity>** which describes different physical influences on the sensor or measurement. This element consists of several **<conditions>** that are described by name, ranges together with a resulting error and a current state. Here it is possible to numerate different working conditions for sensors. For example a humidity sensor could be given a range for a working

temperature and a resulting error. <maintenance> <durability> and <abrasion> elements also describe aspects of cross sensitivity and a resulting error.

The complete extended attribute schema for degradation in the nexus platform is listed in [Köp08]. Further explanations for the terms used can be found in [fMuG95]. A complete list of metrics describing the quality of different kinds of scalar sensor data is described in [Köp09].

3.1.2 Reliable updates for information in environmental models using sensors

The main objective of the Nexus Center of Excellence is the definition and realization of dynamic shared context models for context-aware applications. In this scope, issues concerning communication, information management, methods for model representation and sensor data integration are covered. Based on these digital world models, new innovative applications become possible, which can access information of the real world originating from sensors and additional, aggregated information. We currently witness the rapid proliferation of different kinds of sensor systems. These systems allow the acquisition of context information and make the integration of the sensor data an important research aspect. An open question is, which sensors are suitable for providing context information to the world model with as little redundancy as possible. The problem in updating the world models by sensor measurements is to reduce uncertainty and inconsistency.

If a local world model of an application or agent conflicts with data in the shared context model the system contains an inconsistency. It is hard to decide, whether the data of one single local world model based on sensor data is erroneous or if the shared world model is out of date and needs a correction. If the application needs to be sure about this specific information to work properly, the inconsistency needs to be resolved. One solution to this problem is to repeat the measurement with different sensors and to reduce uncertainty by redundancy. Methods to address other agents with access to sensors and to communicate the sensor data are provided by the Nexus Platform. The study described in this paper examines methods to statistically optimize the reduction of uncertainty where only a few measurements of corresponding physical values are available, which leads to critical and inescapable statistical problems in the rating of the sensors. The reduction of uncertainty is necessary to make relevant contributions to the shared world model, maintaining a high degree of reliability while keeping costs and expenditure of time within an acceptable range by involving only a few number of sensors, services or devices in the process.

This section describes the normalized arithmetic mean algorithm to reliably update and extend environmental models. An evaluation of the algorithm using real sensor data is described in [KBZ⁺08], a more detailed description of the underlying agent negotiation is described in [BKZ⁺09].

Agent Negotiation

If a measurement of an agent's sensor does not fit to the corresponding data in the shared context model the system contains an inconsistency. In this case a negotiation protocol is initiated to resolve the inconsistency using several sensors. The Nexus platform identifies registered mobile or static agents with corresponding sensors. The more sensors and measurements are available, the easier it is to increase the accuracy of the measurement statistically. But to keep the duration and costs to resolve the inconsistency in an acceptable range we assume that only a few number of measurements are available to the platform. Therefore we try to get the best results from only a few measurements by rating each sensor. After the negotiation has been completed the agent can decide whether the reason for the conflict was its own measurement or erroneous data in the shared context model. If necessary it should update the data on the context server and add meta data describing key data from the negotiation as well as the reliabilities of the participating agents. Calculating the weighted arithmetic mean of all the gathered measurements, the reliabilities of the agents and processing the fuzzy clustering is described in detail in the following sections.

Sensor Data Processing

The **statistically optimized** approach is based on statistics considering optimized combination of measurements from the book by Dietrich [Die91]. In use of the open Nexus platform as described above we assume that we can reach only a few agents that are able to execute measurements of one single physical value keeping costs and expenditure of time within an acceptable range. This limitation forbids to rely only on the arithmetic mean of the measurements. One solution to reduce the error in calculating a result from multiple sensor data is to estimate the standard deviation of each sensor, which leads to a maximum likelihood estimator.

A value x is measured by n different sensors giving the results x_1, x_2, \dots, x_n . Assuming the noise in the sensor data is normally distributed and that we know the standard deviations $\sigma_1, \sigma_2, \dots, \sigma_n$, where q is the most likely value of x the deviations of the results are $q - x_1, q - x_2, \dots, q - x_n$. So the combined probability of the deviations is the product

$$\prod_{i=1}^n \left(e^{-\frac{(x_i - q)^2}{2\sigma_i^2}} \right) = \frac{e^{-\sum_{r=1}^n \frac{(x_r - q)^2}{2\sigma_r^2}}}{(2\pi)^{\frac{n}{2}} \prod_1^n \sigma_r}$$

The maximum of the combined probability results in the most likely value for q . This leads to

$$q = \frac{\sum_1^n \frac{x_r}{\sigma_r^2}}{\sum_1^n \frac{1}{\sigma_r^2}} \quad \text{or} \quad \bar{q} = \frac{\sum_1^n \frac{\bar{x}_r m_r}{\sigma_r^2}}{\sum_1^n \frac{m_r}{\sigma_r^2}}$$

3.1 Universal Model for the Quality of Sensor Data

where arithmetic means \bar{x}_r and standard deviations $\bar{\sigma}_r$ are approximated from a limited number m of measurements of the sensors in the past, with

$$\bar{x}_r = \frac{1}{m} \sum_1^m x_{r_i} \quad \text{and} \quad \bar{\sigma}_r = \frac{\sigma_r}{\sqrt{m_r}}$$

To adopt this method to the multiagent system we set

$$\sigma^2 = \sum_1^m \frac{(x_r - \bar{x})^2}{m - 1}$$

and assume $\bar{x} = 0$ which means that the result of negotiations in the past results in a correct value without noise. In fact each result still contains noise if it is not derived from an infinite number of measurements. But the simulations described later in this section with $m > 200$ show that this assumption is acceptable.

Disadvantage of the Statistically Optimized Approach

The described limitations to only a few participating agents can lead to a situation where every negotiation ends in taking the result from the same agent. This happens if the result of the negotiation is incidentally more than once in a row near the measurement of one single agent. In this case the estimated deviations that improve the result of a single negotiation compared to the simple arithmetic mean can run the system into a singular behavior because of the positive feedback from updating the deviations from the result after each negotiation. A problem occurs when one single agent wins a big part of negotiations that are kept in the buffer for the estimation, which is the more likely the fewer agents are participating in the negotiations.

Fig. 3.1 shows the singular behavior of the system in a simulation. The buffer stores 200 negotiation results to estimate the standard deviations which is reflected by the initialization phase visible in the diagram. After several negotiations the standard deviation of a single agent converges to 0. As soon as the standard deviation of one agent is estimated to be 0 the system is not able to leave this state anymore. Each following negotiation of measurements is reduced to an acceptance of the measurement of this agent as a result. Therefore an application of this method in the open nexus platform is not practicable. A complete description of the system and its singular behaviour can be found in [Köp08].

Normalized Weighted Arithmetic Mean

The singular behavior of the above described negotiation method can not be avoided, but an enhancement enables the system to recover from such a situation immediately. The enhancement is based on a preparatory description of the accuracy of each sensor. A restricted quality description of each sensor prevents an infinite weight of a single measurement. To realize the restriction we normalize the standard deviation of a sensor from the range $[0; \infty]$ to a reliability z_r in the range $(0; 1]$ where it is

3 Degradation Model

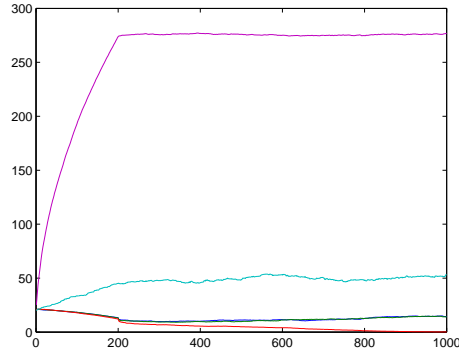


Figure 3.1: Singular behavior: Absolute deviations in a simulation of 1000 negotiations

important to distinguish several good measurements from each other. Therefore we use $f(x) := 1 - e^{-x}$ for the normalization. In a negotiation the weight for each measurement x_r of a sensor is defined by $w_r = 1 - z_r = e^{-x}$. And the combination of several measurements to one result is

$$q = \frac{\sum_1^n w_r x_r}{\sum_1^n w_r}$$

Normalizing

It is necessary to use the same normalization for each sensor with the same standard deviation. Consider a negotiation with three participating agents. Using sensors with identical standard deviations but different normalizations the system again runs into a singular behavior and in the end takes the measurements from the agent that benefits from its normalization. But on the other hand, if we have different types of sensors with unequal standard deviations or resolutions it is not practicable to use the same normalization for the sensors reliabilities. Hence individual normalizing is necessary for each sensor and it has to be calibrated once before the sensor data can be included in negotiation processes.

To realize an individual normalization the corresponding function $f(x)$ has to be modified using a coefficient. This leads to

$$f(x) := 1 - e^{-\gamma x} \quad \text{with} \quad \gamma = -\frac{\ln(\eta)}{\sigma}$$

where σ is the anticipated standard deviation of the sensor and η the desired normalized value. The anticipated standard deviation of the sensor has to be acquired manually by measuring reference values or can be adopted from the sensor's hardware specification (which has to be reliable). We set the target for the normalized value which represents a deviation of a single measurement in the range of the standard deviation to $\eta = 0.5$ which is in the middle of the range of

the weights. The definite value of η can be varied, but it should not be near the boundary of the range of the sensor’s reliability to maintain proper differentiation and avoid numerical problems. The target value η has to be the same for every agent participating in a negotiation process.

Results from Simulated Negotiations - Proving Robustness

The system did not run into a singular behavior itself using the normalized weighted arithmetic mean. To prove the robustness and to test the recovery of the system from disturbances we artificially altered the system. First the weighted deviation of one agent with an offset in the measurement results was set to a very low value which results in a big weight of this agent’s measurements in the following negotiations. The estimation of the standard deviations of the other agents is also influenced. The disturbance was added in negotiation 400 and the modification is shown in Fig. 3.2. This test shows that the system can recover from the state that one agent’s deviation is estimated to be 0 where the system described before ran into the singular behavior. After the number of negotiations corresponding to the buffer size the system has completely recovered. Similar to the behavior in the initialization process.

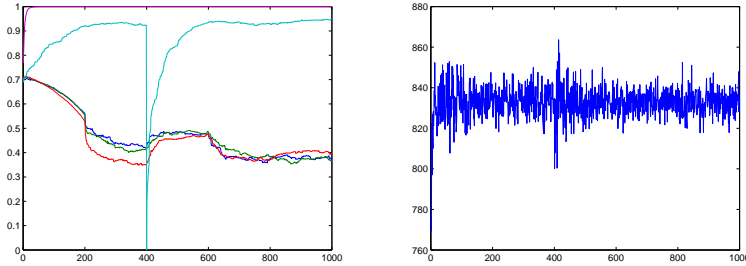


Figure 3.2: 1000 consecutive simulated negotiations with 5 agents. The deviation of one sensor is artificially set to 0 in negotiation 400

The second disturbance to the system is an artificially added noise to the measurements of one sensor. In reality this could be the result of a physical influence to the sensor from which it gets decalibrated. The effect is similar to the one of the first disturbance and shown in Fig. 3.3. One agent overestimates its own reliability after the disturbance. This influences the results of the negotiations and the self-assessment of the other agents. The system adapts to the new conditions during a number of negotiations according to the described buffer size.

Preprocessing to Exclude Erroneous Measurements

Each agent providing sensor data has to execute a preprocessing to improve the quality of the information. The described negotiation process combines the measurements in general to one result without any knowledge about its semantics. But an agent itself can preprocess the sensor data with regards to specific properties of the sensor, e.g. using a low-pass filter. In contrast to the preprocessing by the agent,

3 Degradation Model

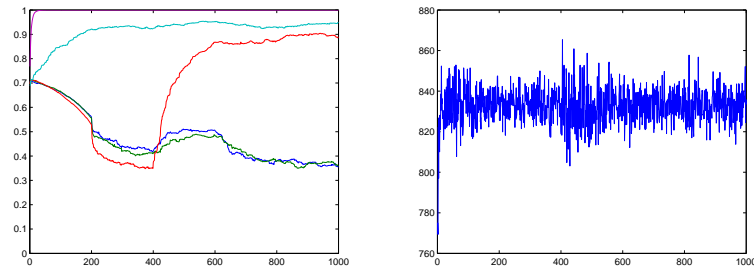


Figure 3.3: 1000 consecutive simulated negotiations with 5 agents. Artificially noise is added to the measurements of one sensor from negotiation 400 onward.

the preprocessing in the negotiation process can compare the measurements from different sensors to the same phenomenon before combining them to one result.

Clustering of the sensor data is one solution to identify and exclude erroneous or at least abnormal measurements from calculating the result [Bac96]. Clustering works fine if we have lots of measurements from which it has to identify outliers. To prevent the preprocessing from excluding non-outliers from only a few measurements we use Fuzzy Clustering [Tim02]. An evaluation of the normalized weighted arithmetic mean using real sensor data and more details to the algorithm are described in [BKZ⁺08].

3.2 Generic Uncertainty Model for Position Information

3.2.1 Introduction

Position information on moving objects such as mobile devices, vehicles, and users as well as stationary objects such as buildings, rooms, and furnishings is an important kind of context information for context-aware applications. The authors of [SBG99] and [DA00] even refer to the locations of objects as *primary context*.

Position information is generally subject to uncertainties at every stage of processing: Already position information acquired by positioning sensors such as GPS receivers only approximates the actual position of the respective sensor or object due to physical limitations and measurement errors of the sensing hardware. Update protocols for transmitting position information from sensors to remote databases or location services further degenerate the position information for the sake of reduced communication cost [LR01, vJP05, LFDR09]. Interpolating in time between consecutive pairs of position records may result in further uncertainties, depending on the temporal density of the position information. Fusion of position information on the same phenomenon improves the accuracy but cannot eliminate uncertainties altogether.

3.2 Generic Uncertainty Model for Position Information

Many context-aware applications must not neglect such uncertainties. For instance, navigating a blind person around obstacles [HKBE07] requires estimates for the uncertainty of the position information about the blind person as well as about the obstacles.

Therefore, a variety of mathematical models for uncertainty of position information have been researched and proposed in the last decades, depending on the specifics and properties of the different technologies and algorithms. For instance, GPS receivers model the distance between sensed and actual position by normal distributions depending on measurement errors and satellite constellation [UN08]. The authors of [PJ99,CKP04] model all possible positions in-between two given position records by intersecting two circles around the positions given in the records, resulting in a lense-shaped area.

Based on these findings, different *uncertainty-aware* interfaces for accessing and querying position information have been proposed for the different system components such as positioning sensors, update protocol endpoints, moving objects databases, and location services.

With the advent of large-scale context-aware systems such as the Nexus platform [LCG⁺09b], applications more and more rely on position information from many different sources. Therefore independent and technology-specific uncertainty models and query interfaces are not sufficient but a generic approach allowing for homogeneous and uncertainty-aware access to position information is required.

Regarding such an approach, we can again distinguish between a generic, mathematically principled model for uncertain position information and a suitable, extended query interface based on this generic uncertainty model. The requirements for the generic uncertainty model are as follows:

- *Expressiveness and generality*: The generic uncertainty model has to be fully compatible with all existing specific uncertainty models for position information of the different positioning sensors, update protocols, and fusion and interpolation algorithms and reflect them with minimal loss of information.
- *Directness*: For simplicity and for minimizing the computational and storage overhead in implementations, the generic uncertainty model has to represent the uncertainty of position information in a self-evident way corresponding to the specific uncertainty models.

The resulting basic requirements for the extended, uncertainty-aware query interface are the following ones:

- *Immediacy and comprehensiveness*: The query interface should immediately build upon the generic uncertainty model to minimize computational effort and exploit all information provided by the uncertainty model.
- *Generality*: The query interface has to provide all prevalent spatial query types for position information such as position query, range query, and next-neighbor query, cf. [GS05].

3 Degradation Model

For position information in context-aware systems, we propose a generic uncertainty model based on partial spatial distribution functions and a corresponding extended query interface supporting five prevalent query types satisfying the above requirements. Our approach is suitable for all (uncertain) point-shaped position information in the two-dimensional space.

In detail, we present the following contributions: In Section 3.2.2, we survey existing, specific uncertainty models for position information. In Section 3.2.3, we show that they can be classified into three fundamental types and that they all base on partial spatial distribution functions and then derive our generic uncertainty model. Based on this finding we present an extended query interface for uncertain position information in Section 3.2.4 and show how to implement this interface for different specific uncertainty models. Section 3.2.5 discusses related work, before we present a summary in Section 3.2.6.

3.2.2 Survey of Uncertainty Models

In this section, we survey the existing uncertainty models for position information. Most of these models only consider two-dimensional positions. Height information of indoor positioning systems is often reduced to information about the floor and, in case of outdoor systems such as GPS, the height information is handled separately. Therefore, we restrict this survey to two-dimensional positions.

3.2.2.1 Specific Uncertainty Models for Position Information

Positioning systems use different techniques like triangulation/-lateration, scene analysis, or proximity sensing to determine positions [HB01]. Different positioning techniques not only yield different scales of accuracy—from millimeters to hundreds of meters—but also result in different uncertainty models.

Positioning systems based on trilateration mostly model the position sensed at time t and denoted by s_t as a normal distribution. This particularly applies to global navigation satellite systems such as GPS [UN08] and ultrasonic-based positioning systems such as Cricket [SBGP04]. For instance, according to [UN08] the standard deviation σ of a two-dimensional position determined by GPS can be calculated based on the User Equivalent Range Error σ_{UERE} and the Horizontal Dilution of Precision, HDOP, as

$$\sigma = \text{HDOP} \cdot \sigma_{\text{UERE}} \quad (3.4)$$

Other systems—e.g., the WiFi positioning system presented in [BP00]—only give a center point and several percentile values around that point, i.e. s_t consists of several concentric circles expressing the probability that the actual position, denoted by a_t , lies within a given circle. In beaconing systems, such as Active Badge [HH94], the sensed position may only specify an area such as a room—i.e. a_t is known to be in that area but without any further distribution information. The same applies to the Smart Floor positioning systems using pressure-sensitive tiles [OA00] and positioning using passive RFID tags [MKT08].

3.2.2.2 Modeling of Uncertainty in Update Protocols

Update protocols introduce further uncertainties to position information and lead to new uncertainty models. Dead reckoning protocols trade uncertainty off against communication cost for efficient transmission of position information from a remote positioning sensor to stationary components managing the current position [LR01, vJP05]. Remote trajectory simplification algorithms additionally consider the costs for storing the current and past positions, i.e. the whole trajectory [LDR08, LFDR09]. For all these approaches, a position s_t is modeled by a center point and a distance value for the maximum distance from that point. However, the distribution within the resulting circle is undefined.

3.2.2.3 Uncertainty Models for Fusion and Interpolation

Fusion algorithms improve the accuracy of a sensed position from different sensor data on the same phenomenon. Multi-Area Probability-based Positioning by Predicates [Rot07] describes s_t by a number of polygons with probability values taking into account even multiple predicates on the position. For fusion of arbitrary probability density functions different Bayes filter implementations are applicable, possibly discretizing the plane using a grid [FHL⁺03].

Complex, uncertainty-aware interpolation algorithms allow for deriving positions at times in-between two sensing operations taking into account the temporal discretization introduced by sensing. The authors of [PJ99, CKP04] show how to restrict the position s_t at such a time to a lense-shaped area—the intersection of two circles—by means of the maximum speed between the sensing operations.

3.2.3 Mathematical Generalization for Time-dependent Point Data

The diversity of these specific models makes it difficult to incorporate uncertain position information from different sources in applications. We show in the next section, however, that all these different models can be reduced to a common mathematical model for uncertain position information. After that, we propose a consistent interface for applications that need to access uncertain position data from different specific models. This interface is based on our common mathematical model.

3.2.3.1 Classification of Uncertainty Models

Although a large number of different specific models for uncertain position information exists, we can classify them into three major types as illustrated in Figure 3.4:

1. *pdf-based models*: These models use complete probability density functions to describe the uncertainties of positions. Hence, with such a model, a position at time t is described by a two-dimensional probability density function $s_t : \mathbb{R}^2 \rightarrow [0, \infty)$.

Amongst others, pdf-based models are used for specifying the uncertainty of trilateration-based positioning systems such as GPS.

3 Degradation Model

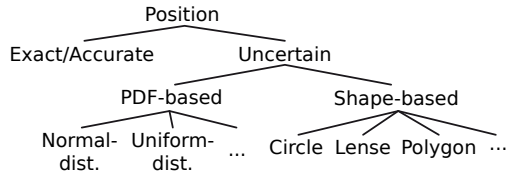


Figure 3.4: Taxonomy of major classes of the existing uncertainty models.

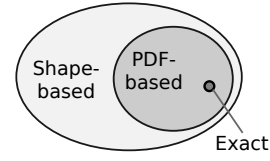


Figure 3.5: Mathematical generality of the major uncertainty models.

2. *Shape-based models*: Models of this class describe positions by geometric shapes. These shapes have associated probability values, however, in contrast to the pdf-based models, the approaches make no claims about the probability distribution within a shape.

Hence, a position at time t is a set $s_t = \{(A_1, p_1), \dots, (A_n, p_n)\}$ where $p_j \in [0, 1]$ and $A_j \subseteq \mathbb{R}^2$ are geometric shapes such as polygons or circles.

Shape-based models are used, for example, for position information from infrared beacons, RFID tags, or interpolation with the intersection of circles.

3. *Accurate model*: For completeness, we also include the accurate model for specifying an exact position without uncertainty.

Formally, a position is described by a single point, representing the actual position, i.e. $s_t = a_t$.

3.2.3.2 Generic Uncertainty Model

In terms of probability theory, all three classes of uncertainty models provide probabilistic information on the actual position they describe as they allow for mapping from one or more geometric shapes A to cumulative probabilities p . More precisely, they describe the position at time t by a (generally partial) function $s_t : \mathcal{P}(\mathbb{R}^2) \rightarrow [0, 1]$ with

$$s_t(A) = P[a_t \in A] = p. \quad (3.5)$$

We refer to such a function as partial spatial distribution function (psdf). Note that pdf-based models even allow for computing a mapping for all $A \in \mathcal{P}(\mathbb{R}^2)$ and thus can be treated as special, non-partial cases of psdf. Accurate positions given by the accurate model likewise are special cases of psdf, where $s_t(A) = 1$ if $a_t \in A$, and otherwise $s_t(A) = 0$.

Thus, regarding psdf, the three classes of uncertainty models can be nested according to their mathematical generality as illustrated in Figure 3.5.

3.2 Generic Uncertainty Model for Position Information

It holds that $A_j \subseteq A_k$ implies $s_t(A_j) \leq s_t(A_k)$ as well as $s_t(\mathbb{R}^2) = 1$. Thus, given an arbitrary area A , a psdf s_t allows for deriving two estimates p_{lower} and p_{upper} with $0 \leq p_{\text{lower}} \leq s_t(A) \leq p_{\text{upper}} \leq 1$ for the position of the corresponding object at time t .

The three major classes of uncertainty models and their common basis in terms of probability theory is an important finding and composes the generic uncertainty model for the extended query interface proposed in the next section.

As the generic uncertainty model includes all classes of specific uncertainty models, it satisfies the requirements *expressiveness* and *generality* given in Section 3.2.1. Furthermore, it satisfies *directness* as it immediately bases on shape-based models, the most general class of uncertainty models in mathematical terms.

3.2.4 Uncertainty-aware Query Interface for Position Information

In this section we present an extended, uncertainty-aware query interface for position information based on the above finding of a generic uncertainty model. Therefore, the query interface can be implemented for every existing uncertainty model and thus allows for uncertainty-aware processing of position information from different, heterogeneous sources.

Next, we discuss the extended, uncertainty-aware versions of prevalent spatial query types for position information and thus show that the query interface meets the requirement *generality* given in Section 3.2.1. Then, we describe a number of examples how to implement the query types and thereby the query interface for different specific uncertainty models, which shows that the query interface also satisfies the requirements *immediacy* and *comprehensiveness*.

3.2.4.1 Extended Query Types

In the following, we consider an arbitrary set of objects $\{O_1, \dots, O_n\}$, moving or stationary. Though most entities of the real-world have a certain extent, we only consider point objects. For any given object, one can always define an anchor point and thus reduce its position to this point. We denote the position of object O_i at time t by $s_{i,t}$. All queries have two parameters O_i and t in common, specifying the queried object and the queried time, respectively.

Besides the uncertain position information, we argue that the providers must define most likely point positions for each time and object they manage. This *defined point* $c_{i,t}$ for an object O_i at time t may be either modeled explicitly or computed on the fly from the uncertain position information of O_i . Note that this point is naturally given with most existing uncertainty models such as normal distributions or circular shapes.

The defined point $c_{i,t}$ of O_i at time t serves to define an unambiguous mapping $\bar{c}_{i,t} : [0, 1] \rightarrow \mathcal{P}(\mathbb{R}^2)$ from each cumulative probability p to the circular area $\bar{c}_{i,t}(p) = A^C$ with center $c_{i,t}$ and minimum radius such that $s_{i,t}(A^C) \geq p$. This is needed, as the inverse $s_{i,t}^{-1}(p)$ is generally ambiguous. For instance consider the 2D normal

3 Degradation Model

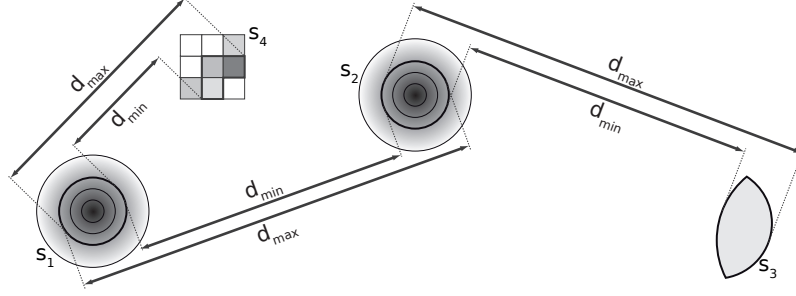


Figure 3.6: Distance query evaluation.

distribution illustrated in Figure 3.7: The left half, the right half, and the inner circle are three examples of areas with $s_{i,t}(A) = 0.5$.

Where applicable, the circular areas $\bar{c}_{i,t}(p)$ are clipped to $A_{i,t}^1$, the smallest area with $s_{i,t}(A_{i,t}^1) = 1$, which always is unambiguous but may be equal to \mathbb{R}^2 .

Position Query Besides O_i and t , the position query takes a parameter $p \in [0, 1]$ and returns the smallest area $A = \bar{c}_{i,t}(p) \cap A_{i,t}^1$ such that $s_{i,t}(A) \geq p$:

$$\text{Position Query: } \text{PQ}(O_i, t, p) \rightarrow (A, \bar{c}_{i,t}(p), s_{i,t}(A)) \quad (3.6)$$

Moreover, it returns $\bar{c}_{i,t}(p)$ and the probability value $s_{i,t}(A)$.

Inside and Range Query To test whether an object is within an area A with a probability of at least $p_{\text{true}} > p_{\text{false}}$, the inside query is defined as:

$$\text{Inside Query: } \text{IQ}(O_i, t, A, p_{\text{true}}, p_{\text{false}}) \rightarrow (\{\text{true, maybe, false}\}) \quad (3.7)$$

With the estimates for p_{lower} and p_{upper} from Section 3.2.3, the inside query returns **true** iff $p_{\text{lower}} \geq p_{\text{true}}$ and **false** iff $p_{\text{upper}} \leq p_{\text{false}}$. In all other cases, the uncertain position obviously overlaps the area A as well as its inverse $\mathbb{R}^2 \setminus A$ and the query returns **maybe**.

The range query can be implemented easily by inside queries on the set of queried objects.

Distance and Nearest-Neighbor Query The distance query returns an upper and lower bound for the distance between two objects with a minimum probability of p by computing the minimum and maximum distances d_{min} and d_{max} between the two shapes $\bar{c}_{i,t}(p) \cap A_{i,t}^1$ and $\bar{c}_{j,t}(p) \cap A_{j,t}^1$.

$$\text{Distance Query: } \text{DQ}(O_i, O_j, t, p) \rightarrow (d_{\text{min}}, d_{\text{max}}) \quad (3.8)$$

Figure 3.6 illustrates several examples of how d_{min} and d_{max} are computed. In Section 3.2.4.2, these examples are discussed in detail.

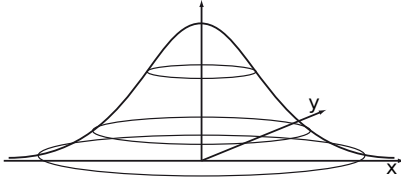


Figure 3.7: 2D normal distribution.

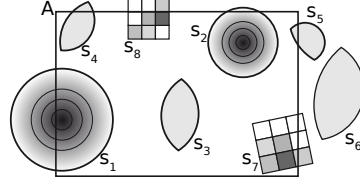


Figure 3.8: Range query evaluation.

The nearest-neighbor query uses these distance bounds to derive the set of objects that may be closest to a given object O_i . Given O_i and a probability value p , it computes the pairwise distances between O_i and all other objects O_j ($i \neq j$) as described above and determines the maximum distance \hat{d} for O_i 's nearest neighbor as $\hat{d} = \min(d_{\max})$ for all $O_j \neq O_i$. Then, it returns the set of objects with their distance bounds that may be closer to O_i than \hat{d} :

Nearest-Neighbor Query: $\text{NNQ}(O_i, t, p) \rightarrow (O_j, d_{\min}, d_{\max})^*$ where $d_{\min} \leq \hat{d}$

Thus, depending on p , the result either contains only objects that are likely to be nearest neighbors or also objects with a low probability of being nearest neighbor.

3.2.4.2 Implementing the Query Interface

In the following, we exemplarily discuss how to implement the proposed query interface for three specific uncertainty models. For actual implementations different spatial data models such as the simple feature types of the Open Geospatial Consortium (OGC) are feasible¹.

As a first example, we consider the uncertainty model of GPS [UN08] based on normal distributions. Then, we discuss the lense-based uncertainty model of the interpolation algorithm in [PJ99, CKP04]. Finally, we show how to map a grid-based uncertainty model [FHL⁺03] to the extended query interface. For all of these models, we show how to implement PQ , IQ , and DQ . We leave out RQ and NNQ since these are straight-forward extensions of IQ and DQ .

GPS Uncertainty Model based on Normal Distribution A GPS position is given by longitude, latitude, and the HDOP value specifying a 2D normal distribution. Longitude and latitude can be directly used as defined point $c_{i,t}$ for the generic model. As already described in Equation 3.4, the HDOP value is multiplied with the device-specific User Equivalent Range Error, σ_{USERE} , to derive the standard deviation σ of the normal distribution [UN08].

As an example consider a GPS sensor with accuracy $\sigma_{\text{USERE}} = 5$ m reporting $(lat, lon, HDOP)$ as $(48^\circ 47' N, 9^\circ 11' O, 1.5)$. In this case, a PQ with $p = 0.75$ returns

¹Depending on the data model, curves (e.g., of circles or lenses) have to be approximated by polygons at a suitable level of granularity.

3 Degradation Model

a circle A^C that is centered at $(48^\circ 47' N, 9^\circ 11' O)$ with radius $r = 8.84$ m by solving the circular integral over the density function $f_{N^2(0,\sigma^2)}(x,y)$ of the two-dimensional normal distribution with $\sigma = \text{HDOP} \cdot \sigma_{\text{URE}} = 1.5 \cdot 5 \text{ m} = 7.5$ m for r , i.e. by solving

$$p = \int_{\sqrt{x^2+y^2} \leq r} f_{N^2(0,\sigma^2)}(x,y) d(x,y) . \quad (3.9)$$

Figure 3.8 shows a queried range A and the positions of eight objects. s_1 and s_2 are GPS positions where an IQ can be unambiguously evaluated by integrating $f_{N^2(0,\sigma^2)}(x,y)$ over A . For $p_{\text{true}} = 0.8$ and $p_{\text{false}} = 0.2$, IQ returns **maybe** for s_1 and **true** for s_2 .

Figure 3.6 shows several positions of objects and the upper and lower bound for the distances between pairs of these positions. In case of a DQ on two GPS positions s_1 and s_2 with $p = 0.75$, A^C is computed for each of these positions according to the explanations for the PQ . The lower bound d_{min} for the distance between the positions is then computed as the minimal distance between the resulting circles. Similarly, the upper bound d_{max} is computed as the maximal distance between these circles.

Lense-based Uncertainty Model For interpolation with lenses [PJ99, CKP04] (cf. Section 3.2.2), we consider two consecutive position fixes of an object O_i at times $t_1 = 0$ s and $t_2 = 100$ s in the Euclidean plane with $s_{i,t_1} = (0 \text{ m}, 0 \text{ m})$ and $s_{i,t_2} = (100 \text{ m}, 0 \text{ m})$. In addition, we assume the maximum speed of the object is known to be 1.5 m/s. The position query PQ for time $t = 50$ s returns the intersection of the circles centered at s_{i,t_1} and s_{i,t_2} with radius $r = 1.5 \text{ m/s} \cdot 50 \text{ s} = 75$ m for any queried p . Note that the probability $s_{i,t}(A)$ given in the query result always is 1.0.

Also note that any point within the lense can be chosen as defined point $c_{i,t}$ without affecting the result $A = \bar{c}_{i,t}(p) \cap A_{i,t}^1$ of the PQ (cf. Equation 3.6). An obvious choice for $c_{i,t}$ is the linear interpolation between s_{i,t_1} and s_{i,t_2} .

For the queried range A given in Figure 3.8, the IQ returns **true** for position information s_3 , **false** for s_6 , and **maybe** for s_4 and s_5 for any value of p specified in the queries.

Figure 3.6 shows an example for a DQ involving a lense-based position s_3 . To evaluate the DQ between s_3 and the GPS position s_2 with $p = 0.75$, a PQ on s_3 is processed, which results in a lense shape. Then, the lower bound d_{min} for the distance between the positions is computed as the minimal distance between the lense shape of s_3 and A^C of s_2 . Similarly, the upper bound d_{max} is computed as the maximal distance.

Grid-based Uncertainty Model Consider a grid-based [FHL⁺03] position that is given by a set of tuples (x_j, y_j, p_j) where x_j, y_j are cell coordinates and p_j is the corresponding probability. Thus, the grid-based position is given by a set of disjoint quadratic shapes with associated probabilities. As defined point $c_{i,t}$ for the generic

3.2 Generic Uncertainty Model for Position Information

uncertainty model, a couple of alternatives are conceivable: First, the center of the cell with highest probability p_j is chosen. Second, the defined point is selected as the centroid.

As an example consider a grid-based position defined by

$$s_{i,t} = (1, 1, 0.15), (2, 1, 0.05), (2, 2, 0.2), (3, 2, 0.5), (3, 3, 0.1) .$$

We compute the centroid (x_c, y_c) of this position as defined point by taking the weighted sum of cell indices over each dimension:

$$\begin{aligned} x_c &= 1 \cdot 0.15 + 2 \cdot (0.05 + 0.2) + 3 \cdot (0.5 + 0.1) = 2.45 \\ y_c &= 1 \cdot (0.15 + 0.05) + 2 \cdot (0.2 + 0.5) + 3 \cdot 0.1 = 1.9 \end{aligned}$$

A PQ with $p = 0.75$ is evaluated by selecting the cells closest to the centroid until the aggregated probability of the cell equals or exceeds 0.75. In this example the polygon enclosing the cells $(2, 1, 0.05)$, $(2, 2, 0.2)$, $(3, 2, 0.5)$ is returned. For $s_{i,t}(A)$, a value of 0.75 is returned, since the sum of these cells' probabilities equals 0.75.

For the queried range A in Figure 3.8, the IQ for the grid-based position information s_7 can be evaluated unambiguously since the range A is aligned to its grid. As the grid of position s_8 is not aligned to the range A , p_{upper} and p_{lower} differ. p_{lower} is the sum of probabilities of the cells that are covered by range A . In contrast, p_{upper} is evaluated as the sum of probabilities of cells that overlap with A .

Figure 3.6 shows an example for a DQ involving a grid-based position s_4 . The processing of the DQ between s_4 and the GPS position s_1 with $p = 0.75$ is based on the result of a PQ on s_4 . With the grid-based uncertainty model, a PQ results in a polygonal area possibly consisting of multiple unconnected parts. The lower bound d_{\min} for the distance is computed as the minimal distance between A^C of s_1 and the nearest part of the area returned by the PQ . The upper bound d_{\max} is computed as the maximal distance to the most distant part.

3.2.5 Related Work

The proposed query interface for uncertain position information and its generic uncertainty model relates to two fields: Models for uncertain spatial data in general and specific approaches for uncertain position information of moving objects.

Pauly and Schneider [PS05] classify the former into models based on rough sets like the Egg-Yolk approach [CG96] and models based on fuzzy sets like the fuzzy spatial data types proposed in [Sch08]. The models particularly define topological predicates for vague spatial regions but do not aim at a generic model integrating the variety of existing uncertainty models.

A variety of algorithms for processing range and next-neighbor queries on uncertainty position information have been proposed in recent years, e.g., [CKP04, TWZC02, YM03]. Most approaches model uncertain positions as circular areas which can be mapped to the proposed generic model. Moreover, they use compatible semantics for query results such as the MAY/MUST semantics for the containment in queried regions proposed in [YM03].

3 Degradation Model

Existing approaches for fusion of position sensor data—particularly Bayesian filtering [FHL⁺03] and inferring from location predicates [Rot07]—are also covered by the generic uncertainty model as discussed in the previous sections.

3.2.6 Summary

We discussed the need for a generic uncertainty model for position information in large-scale context-aware systems and formulated the requirements for a suitable model and uncertainty-aware query interface.

In addition, we surveyed and classified the variety of existing technology-specific uncertainty models and showed that they all can be considered as partial spatial distribution functions (psdf) with respect to their mathematical generality.

Based on this finding, we proposed an extended query interface for position information by extending common query types with information on the position uncertainty. Furthermore, we discussed how to implement these types for certain prevalent uncertainty models. These examples show that the proposed query interface can provide homogeneous access to uncertain position information from different sources and sensors and that the proposed approach meets the various requirements formulated in Section 3.2.1.

Although we only discussed position information, the mathematical approach can be extended easily to scalar data types (e.g., temperature and velocity) as well as data with three or more dimensions. Of course, the query interface has to be adapted to the relevant query types for these data types.

3.3 Uncertainty Model for 3D Geodata

One issue of cross section project “Metrics and Valuation of Context” is to provide concepts and metrics for the quality evaluation of 3D context models. By means of appropriate quality descriptions, the degradation of 3D data can be modeled and made available for uncertainty-aware queries and applications with geographical reference. For instance, quality information of 3D building models is the basis for analyzing the consistency of different building representations.

While in the area of geodesy and geoinformatics a number of metrics and approaches already exists for the evaluation of 2D data [Gle01], the search for appropriate quality descriptions for 3D geometries is still going on [SHF07]. Different applications require different quality criteria. For example, polyhedral error representations are suited for tasks that are based on volumes such as line of sight calculations [FS08]. Other methods apply 3D surface elements in order to determine Euclidean distances as a quality measure through a Least Squares 3D surface matching [AFGS08]. In statistical theory, covariance matrices are a powerful metric to quantify the uncertainty of geometric entities. The matrix dimension depends on the number of parameters which are used to mathematically define the geometric entity. The standard deviations and correlations of these parameters can be extracted from the diagonal and non-diagonal matrix elements, respectively. Usually,

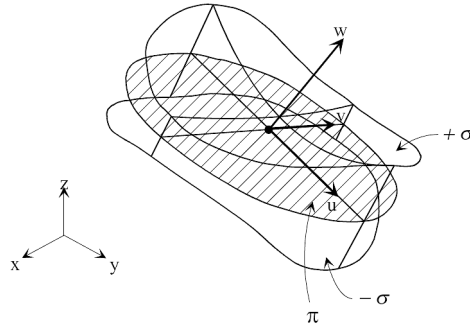


Figure 3.9: Uncertainty region of a plane (from [Haa96]).

the measuring and the generation of basic geometric entities yield uncertainty information as a byproduct. This fact will be utilized for the quality assessment of 3D objects which are relevant for the spatial world model.

One example of 3D data used within Nexus is 3D building models. These are composed of planes specifying the object's surface. Planes are typically derived from at least three 3D points. The uncertainty of a point $\mathbf{p}^T = (x, y, z)^T$ can be represented by a 3×3 covariance matrix with six independent elements. The corresponding uncertainty region is a standard ellipsoid which is defined by the orientations and the lengths of its three semiaxes u , v and w . The orientations are given by the angles $\alpha_x, \alpha_y, \alpha_z$; the lengths correspond to the standard deviations $\sigma_u, \sigma_v, \sigma_w$ in the direction of the respective axis. Consequently, an uncertain 3D point is fully described by the 9-tuple $p : (x, y, z; \alpha_x, \alpha_y, \alpha_z; \sigma_u, \sigma_v, \sigma_w)$. The degradation of a plane estimated from uncertain 3D points can be obtained by error propagation. In Hesse's normal form, a plane π is defined by four parameters a, b, c, d and can be written as $\pi : ax + by + cz - d = 0$ with $d = 1/\sqrt{a^2 + b^2 + c^2}$ and $a^2 + b^2 + c^2 \neq 0$. If only the normal vector $\mathbf{n}^T = 1/\sqrt{a^2 + b^2 + c^2} \cdot (a, b, c)^T$ is uncertain, the error figure of the plane is an elliptic cone through a point $(x_0, y_0, z_0) \in \pi$. When additionally the distance d of the plane from the origin is degraded, the uncertainty region becomes an elliptic bipartite hyperboloid (see Figure 3.9). Relating to an appropriate (u, v, w) -coordinate system, its mathematical form is given by $w^2 = \sigma_w^2 = \sigma_d^2 + \sigma_\alpha^2 u^2 + \sigma_\beta^2 v^2$, where σ_α and σ_β describe the maximum and minimum slope error and σ_d represents the standard deviation of parameter d . Thus, an uncertain plane with the center of gravity (x_0, y_0, z_0) is defined by the 9-tuple $\pi : (x_0, y_0, z_0; \alpha_x, \alpha_y, \alpha_z; \sigma_\alpha, \sigma_\beta, \sigma_d)$ with $\alpha_x, \alpha_y, \alpha_z$ denoting the three angles for the major axis of the hyperboloid. The uncertainty parameters $\sigma_\alpha, \sigma_\beta$ and σ_d can be calculated from the eigenvalues of the plane's 4×4 covariance matrix resulting from error propagation or the plane estimation process itself [För92].

This uncertainty information can now be used for location based applications in Nexus. In the following, three exemplary use cases are introduced. The first one is an inside query of the type "Is person X in building A ?". To answer this question, the uncertainty of both the 3D building model and the person's position have to be

3 Degradation Model

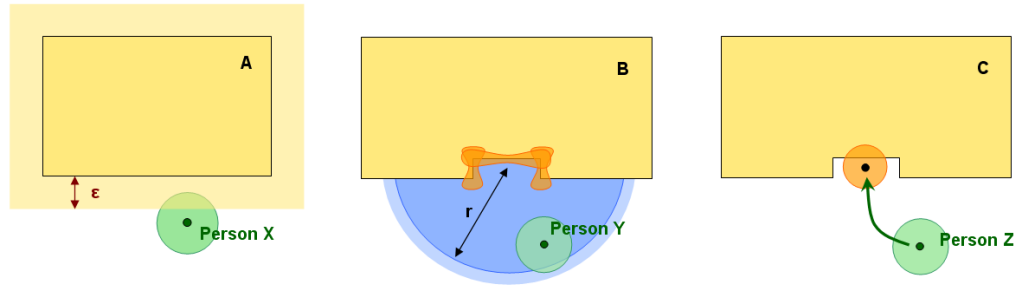


Figure 3.10: Exemplary Nexus applications using uncertainty information of a 3D building model. *Left*: Inside query based on a global uncertainty value of the building. *Middle*: Range query based on local uncertainty measures. *Right*: Navigation task based on local error descriptions.

considered. In contrast to person X that can be treated geometrically as a point object, the significant spatial extent of building A has to be taken into account. The uncertainty of the whole building can be obtained by propagating the errors of its surface planes. The result is a complex uncertainty figure which is difficult to describe mathematically and, thus, hard to handle. However, in cases where such a detailed error representation for the whole building is not required, it may be reasonable to approximate the intricate uncertainty figure by a simple buffer area dilating the building perpendicular to all planes. Here, a threshold ϵ could be used representing a global uncertainty value of the building. Figure 3.10 (left) illustrates this situation in 2D as a projection on the ground.

The second example is a range query referring to specific building parts, as for instance “*Is person Y in the entrance area of building B ?*”. The range in question is the area in front of the entrance door. It can be specified by a circle with a predefined radius r . Considering reconstruction errors of the door, the circle has to be enlarged. For this purpose, the uncertainties of the planes defining the 3D object *door* are analyzed and propagated to the entrance area (see Figure 3.10, middle).

The third uncertainty-aware application is a navigation task, for example, the navigation of person Z to the entrance of building C (see Figure 3.10, right). Similar to the range query, as discussed before, the uncertainty of the door object must be taken into account. However, when the uncertain planes are used to derive the center of the door area, the problem is reduced to a task where only uncertain positions have to be evaluated.

3.4 Uncertainty-Aware Situation Detection with Bayesian Networks

3.4.1 Introduction

In the domain of context reasoning the determination of high level context (hereafter also called as “situation”) is usually done on the basis of environment models that can

be shared by multiple applications [CFJ04] [RAMC04] [GPZ04] [WZGP04]. These are typically ontology based techniques or methods which interferes the context by predefined user-specific rules or pattern. A high level situation recognition process which bases on constraints and correlation of multiple real world sensor data, for example a position information of a GPS receiver, is generally subject to uncertainties. In contrast to approaches in the domain of context modeling, such as [HIR02], most of the methods described in the literature take the degradation [DKN⁺06] of contextual information not consistently into account [GPZ04]. Furthermore all these related approaches base on spatially highly restricted environments and do the reasoning-process in a central component. Accordingly no aspects of distributed systems or positioning strategies has been developed to improve the situation recognition process, determining quality aspects, improve the accuracy or optimize the system load due to positioning information.

Thus we developed a new situation-recognition method, which enables the reasoning process in a general and distributed way. Furthermore the system is able to handle with uncertainties during the situation detection process and even an automatic adaption of uncertainty-aspects during runtime is possible. Therefore a supervised-learning method has been included using only a simple boolean feedback-information of the user. During the implementation different procedures for the recognition of situations were examined and developed based on uncertain context data gained from the *Nexus-Context-Servers (NCS)*.

Due to the novelty a very special focus was on creating a general approach to use common methods for a large set of different situations. During this work two concrete problems within this domain became clear: On the one hand the possibility of uncertain context data due to poor sensor quality, and on the other hand the uncertainty of the actual inference of the situation recognition, for instance due to an incorrect recognition model. To solve this problem we developed an method which uses two different sequential running adaption-steps, which refines the model to improve the recognition process iteratively.

3.4.2 Situation Template Model

To reduce the complexity of the situation recognition process, most context reasoning systems [CFJ04], [WZGP04], [HIR02] use ontology- or predefined rule-based approaches. Another advantage of this proof is the user-friendly and simple designing of rules and models, because of the affinity to human ways of clustering information. In contrary to most of the existing context aware systems which are supposed to cover only a limited geographical area or support only a specific use case scenario [BMK⁺00], [PL03], our focus was on the development of a novel distributable and general situation recognition approach to detect a wide spread of different situations and scenarios. In our approach the coherences and operators as well as the constraints of the different context data are modeled before runtime, based on the implicit knowledge of an expert.

3 Degradation Model

Furthermore we define a situation template as an abstract, machine readable model of a certain basic situation type which could be used by different applications to detect their situation. Thereby the situation template consists of a composition of relations, constraints and coherences of (high-level) context-data. As a consequence a situation template indicates a form of explicit knowledge.

To cover the system requirements, we identified the following necessity of our new recognition model:

- *Generality* : To cover a wide range of possible application scenarios, the template and its representation has to be in an abstract and general way. Both the generality of the model and the generality of the corresponding inference method is to considered there.
- *Distributable* : Due to the idea of the nexus-project, the recognition process has to be distributed in the infrastructure. Thus the model and the corresponding inference process should support the possibility of a distribution and handle with the aspects of a distributed system with the resulting assets and drawbacks too.
- *Uncertainty-awareness* : Because of the use of real sensors, which underlie specific uncertainties, tolerances and latencies, the recognition-process above the sensors-layer has to support accordant metrics and propagation-processes to handle these uncertainties and aggregate these to the application to evaluate their situation.
- *Modularity and Extensibility* : Due to the desired diversity and universality of the recognition process the model should support corresponding interchangeability, expandability and modularity. According to this, the model and even the corresponding inference mechanism has to support the use of several metrics of uncertainty depending on the needs of the application.

According to these necessities a set of different models, techniques and inference approaches has been analyzed and evaluated. Due to the advantage properties of the model we implemented an approach based on probability networks here.

Template structure The representation of a situation-template, based on a probability network, can be modeled as a directed graph, which has a special polytree structure. Because of this polytree structure, the graph is called *Situation Aggregation Tree (SAT)*. Thus in more detail the graph is defined mathematically as $SAT = (O, E)$, where $O = o_1, o_2, \dots, o_n$ represents a set of operators. Furthermore the operators can be differentiated into two different types, the so called *constraint-validators* $CV = cv_1, cv_2, \dots, cv_n$ and the *subsituations* $SN = sn_1, sn_2, \dots, sn_n$.

We define **constraint-validators** as operators which checks whether their input-value (i.e. sensor value) satisfies a predefined condition (using a specific sigmoid operator-function f).

We define **subsituations** as operators that aggregate several operators in one higher aggregated (subsituation-) operator. As a consequence of the template-structure all paths and operators are aggregated in one single top operator. That means, if we want to evaluate a situation-template, we first have to start a reasoning-process and finally check the results of the single top operator.

Furthermore $E = e_1, e_2, \dots, e_n$ represents a set of directed edges or links between the operators which represents the relations between them. Furthermore $e_i = (o_k, o_j)$ is a directed edge from operator o_k to operator o_j . In other words, o_k is the parent-operator of o_j , thus we will say formally $pa(o_j) = o_k$.

Uncertainty metrics As already mentioned above, our reasoning-approach bases on probabilistic networks where we furthermore distinct between two different uncertainty-metrics, similar to other existing reasoning approaches [DA00,MYCD]. One advantage of such a distinction is the resulting modularity, the replaceability of the inference mechanism below and the possibility to expand the uncertainty metric for new ones. This is why we draw a distinction between the *probability-* and the *confidence-metrics* for uncertainty.

- **probability (P):** The probability-metric is defined as a value in the range of 0..1, which represents the probability of the occurrence of the situation from the recognition-process view. For example a probability value of 0.8 means that the situation-recognition-process assumes that the situation is occurring with a probability of 80 percent.
- **confidence (C):** The confidence-metric is defined as a normalized value, which reflects the quality or the correctness of the used situation-template. For example a confidence value of 0.99 means, that the recognition-process will detect the situation (“occurrence” or “nonoccurrence”) with a very high correctness. In contrast a confidence of 0.1 means that the quality of the template (or the underneath context-data) is poor.

Furthermore we differ between these two metrics, because of the desired generality and the possibility of different applications and their individual handling of the confidence. E.g. a high-security-application which controls the access to a banks vault should insist on a high confidence to prevent *false-positive-results*. On the other hand a navigation-application, which warns a blind person not to come across an obstacle, could handle with less confidence (in return with a faster recognition) because the blind person can put up with false-alarm with less consequences.

According the idea of the nexus-project the information of each uncertainty-metric of every single context data can achieved from the *Nexus-Context-Server* initially. Nevertheless it should be clear, that these uncertainties can also be simply obtained from the sensors technical documentation or it can be determined by comparing the sensors observations through training and statistical calculations. For a more detail view, how the nexus framework determine the uncertainty metrics of the sensors, we refer back to the chapter 3.2.

Template representation During the inference the according metric of each used context-data is combined with the corresponding uncertainty metric of the operators in the SAT using Bayesian methods. While this process, the cumulative uncertainty-values are propagated from the bottom to the top of the SAT to evaluate the situation. That means, the uncertainty of the context data is derived from the (physical) sensor, next the constraint-validator uncertainty is derived from the context data, and finally the subsituation-operators uncertainty is determined by aggregating the uncertainties of its parents-operators. Because of the SAT structure and the stochastic independence of the context data, the uncertainty-metrics are propagated in the tree structure bottom-up to the top node. To aggregate the uncertainties during evaluation a set of probability distributions has to be added for each uncertainty metric to the structure of the current SAT.

So we get the new extended structure (see also figure 3.11):

$$\begin{aligned} SAT &= (O, E, \Theta, \Gamma) \\ \Theta &= \{CPT_{\Theta}(o_1), \dots, CPT_{\Theta}(o_n)\} \\ \Gamma &= \{CPT_{\Gamma}(o_1), \dots, CPT_{\Gamma}(o_n)\} \end{aligned}$$

Whereas Θ stands for the *probability-metric* and accordingly Γ stand for the *confidence-metric*. Furthermore $CPT(o_i)$ equates to the *Conditional Probability Table (CPT)* of operator o_i . In detail the $CPT(o_i)$ makes assertions about the according conditional probability of the probability-metric (CPT_{Θ}) and the confidence-metric (CPT_{Γ}) of operator o_i in correspondence to its parent $pa(o_i)$. Using this extension a Bayesian Belief approach is implemented.

The situation-template itself is stored in the template repository in a XML-structure, exploiting the tree-structure of the SAT. For a fast and simple designing of situation-templates and further evaluation purposes we implemented a simple prototype (*Template Designer*) as shown in figure 3.12

To determine the total uncertainty-metrics of the templates we use a common reasoning-mechanism. This reasoning-mechanisms, i.e. the calculation of the uncertainties of the actual situation-template (here: *probability P* and *confidence C*) is done by exploiting the polytree-structure of the situation templates, using the Bayesian Theorem and the *message-passing-algorithm* of Pearl [Pea88], which realize the distributed belief propagation in Bayesian Networks based on Θ and Γ of the according operators.

3.4.3 Parameter-Adaption

For our novel adaption approach we developed an iterative estimation algorithm. Thus the overall method consists of two iteratively running processes:

Step 1: *adaption of metrics* and step 2: *adaption of ontology*.

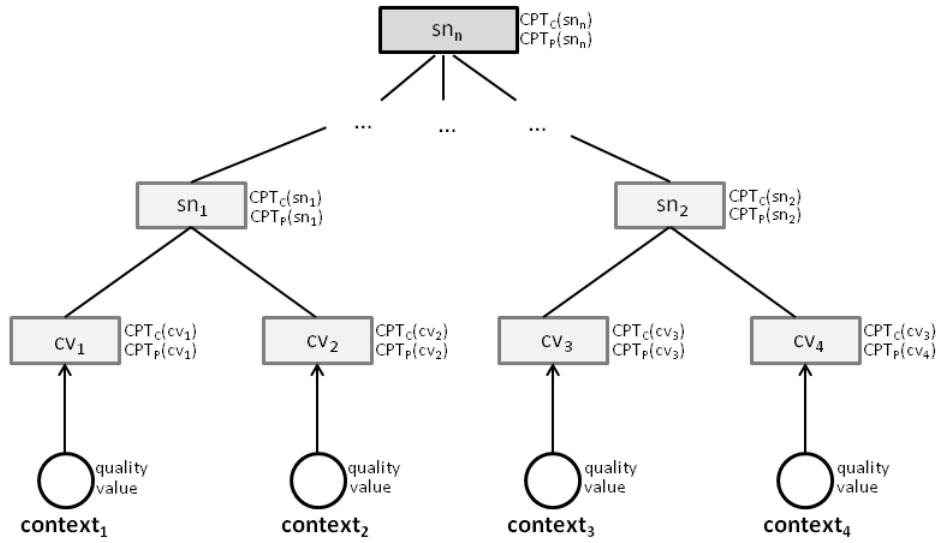


Figure 3.11: Situation Template: SAT-Structure.

Thus the algorithm iteratively adjusts the uncertainty-metric based on the current ontology (= *adaption of metrics*-step) and in the next step the ontology is adapted based on the (new) uncertainty-metric of the first step (= *adaption of ontology*-step) and so on. The algorithm terminates appropriately when a local optimum has been reached.

3.4.3.1 Adaption of metrics

As mentioned above the reasoning-method for situation-detection is done by a Bayesian inference approach using the $CPT_X(o_i)$ of node o_i and the according metric $X \in \{\Theta, \Gamma\}$ for all $o_i \in O$. In this chapter we want to show the method how to adapt these metrics, based on the assumption that the ontology is correct - i.e. without any design-errors or constraint-errors. Thus we are able to adapt automatically the uncertainty-parameters of the operators in the situation template we are using for reasoning. In statistics, many general inference techniques [Tan93], [KR95] have been developed that have been applied to learning of probabilistic networks. Because we assume that the constraints and the ontology are correct we examine the param-

3 Degradation Model

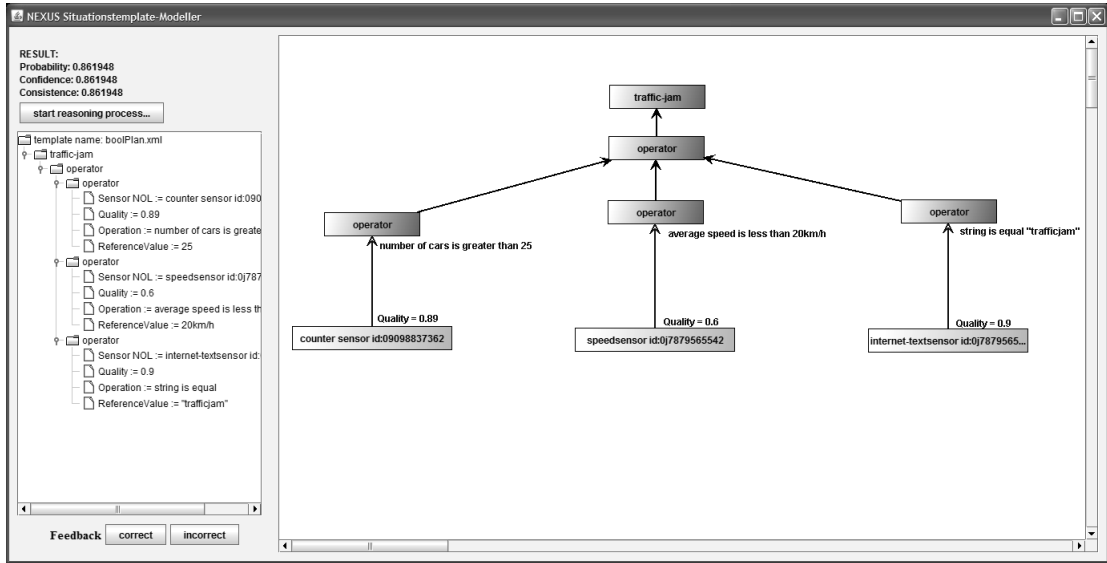


Figure 3.12: Screenshot of Template Designer.

eter fitting problem, i.e. learn the parameters from data. These fitting algorithms exist for probability network (e.g. Bayesian networks) in the cases of complete and missing data [Edw90], [JP95], [Lau95]. Because of the existing uncertainty and the non-observability in our real world domain, we will only consider methods which can deal with missing or incomplete data. Therefore the *Expectation-Maximization (EM) algorithm* [Lau95], [DLR⁺77] is a common and proven approach for dealing with incomplete information when building statistical models. Other common techniques in this domain are the iterative proportional fitting algorithm [JP95] and approaches using evolutionary algorithms [MLD99], [TLS01]. Despite these approaches we use another technique which allows the automatic adaption only with a simple boolean feedback-information based on supervised-learning-methods and even exploits the tree-structure of our SAT. Thus we are using the existing overlaps between learning of Bayesian Networks and Neural Networks, described in [Nea92], [SJJ96], [Bun94].

As already mentioned, each metric $X \in \{\Theta, \Gamma\}$ is represented via an individual and operator specific Conditional Probability Table. These CPTs are predefined by an expert or they can be set randomly initially. We assume that after each reasoning-process the system gets a trustful boolean user-feedback which determine, whether the situation actual occurred or not occurred. With this simple feedback we are able to adapt the uncertainty-metrics in the following way:

During the adaption-step we transform the SAT into a neural net (NN) and interpret each CPT entry as a weight of an according edge of the NN introduced in the work of [RM96]. Furthermore we use a Backpropagation-Algorithm to teach the weights of the NN. For each metric we use an own NN because the CPTs, the input-values and the teaching-vector is different for each metric. I.e. to adapt the

probability-metric we use as input-values for the NN the context-probability-values given by the nexus-platform. As teaching-vector for the NN we use the actual boolean user-feedback, whether the situation occurred or not. Thus we are able to adapt the weights iterative using a simple NN-Backpropagation-Algorithm according to the given context-data (input-value(s) of the NN) and the information of the actual occurrence of the situation (output-value of NN). Furthermore to adapt the *confidence-metric* we use instead as input-values of the NN the context-quality provided by the Nexus-Context-Server. As teaching-vector we use a statistical measurement of the correctness of the corresponding template based on multiple training instances by summarizing the numbers of true-positives and true-negatives results divided by the number of all reasoning-processes. After the optimal adaption of the weights of the NN we transform the adapted neural network back to the SAT-structure, by interpreting the weights of the NN to the corresponding CPT-entry.

3.4.3.2 Adaption of ontology

In this section we want to give an overview of our approach to adapt the specific operator-function of each constraint-validator. This step is necessary because the situation-template is initially designed by a human expert and thus we have to cope with ontology and design errors. In order to be able to correct ontology and constraint errors, they have to be located in the ontology first. One possibility of locating errors is to accomplish data by new measurements several times until the fault location can be clearly determined. Corresponding work can be found in [DKW87], [Rei87], and [Hou94]. In [MRF04] an approach for recognizing and predicting context by learning from user behavior is described.

In one of our previous work [ZHKL09], an algorithm (*Template Adaption Algorithm (TAA)*) for ontology based learning is presented which basic principles are shown next. We assume that the expert already specified a situation template (e.g. using the “Template Designer”). Furthermore we assume that the uncertainty metrics of each operator are already adapted and filled up with the corresponding values in the previous step.

In the next step we create a table, called *Global Control Table (GCT)*. This GCT consists of entries which represents discrete switching states of the constraint-validator function f , as well as the probability value p of each value combination. The probability value p of the according switching combination can be achieved using a simple mapping algorithm based on the confidence-metric of the constraint-validators. For unknown switching combinations or if there are no confidence values available, random values or the value of $p = 0.5$ can be used instead.

For the learning-step we use a supervised learning approach based on the same feedback modul already uses in the adaption step for the metrics. Using this simple boolean feedback-information of the user, the probability value p_i of row i of the GCT depending on the switching state, can be updated according to a simple Delta- Δ -Rule in every recognition step (episode). It appears that for faulty template values and small enough Δ the probability p_i converts with a growing

3 Degradation Model

number of episodes towards 0.

Formally: $\lim_{\#episodes \rightarrow \infty} p_i = 0$.

For every p_i which is under a predefined threshold ϵ , the corresponding row i in the GCT is selected. Afterwards for each of the selected rows a new vector with the switching status will be created. Every created vector represents a potentially faulty template value.

Afterwards statistics, counting the number of all switching states in the vectors is used. For that we define a function $\psi(value)_{cv_i}$ accordingly. Furthermore a helping function $numSet(X, value)$ is defined. This function returns the number of elements $x \in X$ which has the given switching state $value$. We can say formally:

$$\begin{aligned} numSet(X, value) &= \{x | (valueOf(x) = value) AND (x \in X)\} \\ \psi(value)_{cv_i} &= |numSet(cv_i, value)| \end{aligned}$$

In order to be able to correct errors, we first locate the potential error using the previously calculated statistics. It is clear that the vector with the highest value of same switching states in the statistics ψ represents the constraint-validator cv_j which has to be corrected. Afterwards this (faulty) sigmoid constraint-validator-function f will be adapted by shifting the inflection point of f for a predefined factor d (random or fixed value). The result is the automatic correction of wrong or imprecise constraint-validators over n learning episodes. In case that $m > 1$ vectors ($cv_j..cv_k$) have the same maximum amount of switching states compared to another vector two cases have to be distinguished.

- Case 1: The complete set of m selected template values is faulty.
- Case 2: The error is caused by a faulty operator (subsituation which aggregates these m constraint-validators) or by a faulty network-connection.

To detect and distinguish the two error cases a resolving strategy was developed: At first a random context-data value (of cv_i) within the m selected vectors is chosen and updated with the previously described Delta- Δ -Rule. If the statistics of the adjusted vector i improves, the process is continued gradually with the remaining $m - 1$ vectors. If the statistics remains unchanged the error is located in one of the operators which joins the m underlying constraint-validators.

Improvement The basic idea of the improvement of the above described algorithm is not to change the template values with a fixed value or a random factor but change them selectively. For that extension a new table T_{cv_i} has to be created for every constraint-validator. In the case of n constraint-validators a set of n tables T_{vc_i} for $i=1..n$ have to be created. The newly added tables include as columns the calculated global binarized probability-value P of the situation recognition process called *result*, the binary value of the feedback of the user called *feedback* and

the context-data (input of the constraint-validator) queried from the context server called *value*. See also table 3.1 for an simple example.

row i	result	feedback	value
1	1	0	16
2	0	0	15
3	0	1	17
4	1	1	18

Table 3.1: Example of an extension table T_{cv_i} with fictitious context data.

Furthermore it has to be distinguished between the amount of so called “Error Cases” (EC) and “Non Error Cases” (NEC). These are defined as:

$$\begin{aligned}
 EC &= \{row_i \in T_{cv_i} | (result_i XOR feedback_i) == true\} \\
 NEC &= \{row_i \in T_{cv_i} | (result_i XOR feedback_i) == false\}
 \end{aligned}$$

In order to adapt the values in the template a case distinction is necessary. This fact is presented with the example of the operator-function *f* equates *greater-than* “>” -function.

Example: constraint-validator function “greater-than” “>”

- Case 1: maximum value is unequal to the current template value $\in EC$.
 \rightarrow The template value is updated using the value with the smallest difference to the value $V = \max(NEC)$.
- Case 2: maximum value is equal to the current template value $\in NEC$.
 \rightarrow The template value is updated using the value with the smallest difference to the value $V = \min(NEC)$.

3.4.4 Summary

We gave a short survey about the existing methods and approaches of situation detection in context-aware systems. Furthermore we discussed the need of a generic, distributed situation recognition process in a large-scale context-aware system and formulated the requirements of such a system. Based on these necessities we introduced an abstract, machine readable descriptions of a certain basic situation type which could be used by different applications to evaluate their situation, called situation template. Furthermore we introduced the data-structure of the template and the advantages to exploit during the reasoning process on uncertain information. Due to the desired modularity and extensibility of the uncertainty-metric we consider different kinds of uncertainty, where we introduced the uncertainty-metrics Probability and Confidence. Finally we gave an overview of our iterative novel adaption process, which refines the parameters of the uncertainty-metrics in the situation template, to get a more robust and reliable situation recognition system.

3.5 Degradation of high-level context derived from sensor data

Situation recognition based on context information is described for example in [Mie03] and [Fre92] where situations are predefined by conditions. Situation recognition in general combines learned knowledge and observation of the environment. A matching algorithm has to determine which predefined situation fits best to the current state of the world or to the environment of an application or its user. The learned knowledge in this case can consist of situation templates [ZKLL06] that are configured by users or application designers or it can be the result of learning algorithms that cluster context information.

Situation templates are arranged in situation libraries for different context aware applications. It is common that situations concerning the same use case have overlapping preconditions or are correlated with the same phenomenon in the environment but for each situation template with different attributes. To detect a meeting in a room for example a noise level could be used. In this example the same phenomenon would be used for the detection of a meeting where the noise level should be above a defined threshold or for the detection of a situation where some people work in a room and the noise level should be below a defined threshold. To get a more reliable system the situation detection normally is done via bayesian networks which do not only rely on the noise information itself but additionally on corresponding probabilities. Especially when the situation recognition itself should output a quality rating to the determined situation it is necessary that all the analyzed information to the preconditions are rated with qualities or probabilities. A situation recognition that provides probabilities to each possible situation therefore needs probabilities to each information that is assigned to a precondition. This leads to the need of a conversion from uncertainty to probabilities when using sensor data to observe the environment to recognize situations.

In this section we describe the method for deriving high-level context together with a quality measurement from sensor data using logistic regression. An evaluation of the method and a comparison to other methods using neural networks or an empiric approach is described in [KGS⁺09].

Augmented World Model, Degradation and Stochastic Errors

The Nexus platform [NGS⁺01] provides access to context which is managed in distributed augmented environmental models. To extend the environmental models or to update the models to adapt them to the current state of the real world not only user input is used but also sensor data. The platform provides services for reliable sensor data integration by *SensorContextServers* which offer raw and processed sensor data together with ratings such as relative, absolute or standard deviations to context aware applications.

Uncertainty of sensor data is represented in meta data which is divided into several domains of degradation [Köp08] concerning different aspects of quality like temporal

3.5 Degradation of high-level context derived from sensor data

aspects, cross sensitivity or stochastic errors. Since in the nexus platform sensors can be used for different applications. Therefore there is no simple possibility to rate the quality of measurements since quality ratings always have to be related to certain requirements. On a *SensorContextServer* the quality of a measurement is rated in relation to the physical attributes of a sensor itself as long as there is no other specification from an application. For example when a physical sensor has a sampling rate of 10 Hz but the value for applications provided by the *SensorContextServer* is updated only once a second then the quality rating concerning timeliness is 10% or 0,1 respectively.

Since each application can have different quality requirements the *SensorContextServer* can manage weights for particular domains of degradation or complete rating specifications based on physical attributes for each application. An application can for example give higher rates for timeliness instead of accuracy to be able to react fast on changes in the environment. The advantage of this quality management by the platform is to shift the effort of quality monitoring from applications on devices to the servers.

Nevertheless we still have only quality ratings for measurements and no probabilities of information that can be used in a situation recognition. A conversion from these ratings to probabilities is needed on the sensor data level. The goal is to obtain information from sensors for preconditions of situation templates. This means for the situation recognition that each single sensor observing phenomena related to a situation is used individually to give a probability to the relevant situation. These probabilities are combined to obtain a reliable situation recognition using all defined preconditions.

Conversion from raw Sensor Data to Probabilities

To obtain more reliable results for situations which are not directly related to one single phenomenon in the environment of a user or application we define periods of time in which the related phenomenon is observed.

A **Logistic regression** [HL04] is used to determine the probability for an occurrence of an event. Adapted to a situation recognition this can be used to determine probabilities for situations or probabilities for single preconditions of situation templates by learning the assignment of reference measurements to known predefined situations.

As input to a binary logistic regression we used the same information from the periods of time as we used for the neural network approach. The logistic regression learns weights called *regression coefficients* by a maximum likelihood estimation to all the input variables corresponding to the derived information of each period of time. The learning is based on reference measurements where the outcome of the logistic regression is already known. In practice a test set of measurements has to be generated where each measurement can be assigned to one of the situation templates manually. A linear combination of the measurements x_j and the corresponding weights β_j is given in equation (3.10). A situation is assigned to the result z or underlying measurement respectively when $z > 0$. To obtain a quality rating to

3 Degradation Model

the determined situation z is normalized to the interval $]0; 1[$ as shown in equation (3.11).

$$z = \beta_0 + \sum_{j=1}^n \beta_j * x_j \quad (3.10)$$

$$p(y = 1) = \frac{1}{1 + e^{-z}} \quad (3.11)$$

The binary logistic regression is only suitable for distinguishing two mutually exclusive situations or for determining if one single situation is valid or not. In these cases the sum of the according possibilities calculated as described above is one. To distinguish between more than two situations a Multinomial Logistic Regression is necessary [Men01]. The adapted calculation of all the probabilities for n situations is given in equation 3.12 which again normalizes the probabilities to a sum of one.

$$\ln \frac{P(y_i=m)}{P(y_i=1)} = \beta_{0m} + \sum_{k=1}^n \beta_{mk} x_{ik} = Z_{mi} \quad (3.12)$$

The more reference measurements are available the more accurate is the situation recognition in the end. Several quality criterions such as Nagelkerke R^2 [Nag92] can be used to check the quality of the regression with the learned coefficients by applying the system to the reference measurements again. This can be used as a general quality monitoring for situation libraries adapted to individual use cases. The corresponding quality is a hint for application designers or applications that automatically adapt situation recognition to reference measurements stating the need for additional reference information to learn the parameters for the algorithm more precisely.

The ratings for our *meeting* example lead to the usage of the difference of a maximum and minimum measurement and the average of the period of time. The maximum and minimum values themselves were excluded automatically from the situation recognition in our example presented in the next chapter.

Adoption to new situations

The logistic regression system has to be adopted and trained with new coefficients whenever a new sensor is involved or new situations have to be recognized. The advantage is that an application designer doesn't have to specify the ranges of values from measurement results that belong to each new situation. It is only necessary to assign the sensors to known situations. The disadvantage is that the systems needs several reference measurements for each new situation to be trained. But once the reference measurements are available the coefficients can be trained in short time. For our example of 3 Situations and more than 50000 measurements the coefficients could be trained in less than 5 seconds on a normal pc. The different training methods *Enter*, *Forward selection* and *Backward Selection* did not make any difference in the result. All trained coefficients lead exactly to the same situation recognition probabilities.

3.5 Degradation of high-level context derived from sensor data

To execute the situation recognition on different devices the trained coefficients could be provided by the nexus platform. This enables a distributed situation recognition where each sensor's measurements can be assigned to preconditions of situations on the device the sensor is attached to. Afterwards only the calculated probabilities have to be communicated over the network.

4 Consistency Model

Inconsistency in the context of the Nexus project can be observed on the context information layer and on the high level context layer. Inconsistency on the context information layer is caused by different data providers offering different values for the same attribute, while on the high level context layer, inconsistencies are caused by different situation templates for the same situation, which generate different results. In the first case (Section 4.1), the inconsistencies are used as input for the reference model, in the latter case (Section 4.2), inconsistencies are used in refining the situation recognition process.

4.1 Inconsistency on the Context Information Layer

Inconsistency on the context information layer refers to the case that different data providers can offer the same datum, i.e., provide different values for the same attribute. Examples for such situations are different sensors measuring the same datum, or buildings in a town which are modelled by different organizations. In consequence, we have to deal with a finite number of alternative values for the same attribute. The Augmented World Model (AWM) already has the ability to *represent* alternative values, so here we focus on models to *measure* such inconsistencies.

The model used for measuring consistency heavily depends on the types of the attributes (more precisely, the types of the attribute parts) and the model used for representing uncertain values. We have investigated four different domains: The *discrete domain*, containing types like integers, booleans or enumeration values, the *continuous domain with pdf* containing types like real numbers or positions with degradation represented by a probability density function (pdf), the *continuous domain without pdf* consisting of types with degradation represented as described in Section 3 and the *3D-domain* used for three-dimensional building models.

The model for the first three domains has some basic properties, which have been identified as beneficial for applications:

- The consistency is a value from $[0, 1]$ (a pair of values for the generic spatial domain)
- Identical values have the best consistency (1)
- Contrary values have the worst consistency (0). Contrary values are differing certain values or uncertain values, which do not overlap.
- There are cases where two uncertain values are neither identical nor contrary. In this cases, the consistency value is between 0 and 1.

4 Consistency Model

The basic design idea of the model is to estimate the probability that the representation of one provider does not conflict with the representation of the other provider.

The properties of the model for the 3D-domain differ a little bit because of the specialized application domain, but also results in consistency values from $[0, 1]$, where a larger number means better consistency.

In the following discussion, o, o_1, o_2, \dots denote objects. Objects are sets of attributes. The A attribute of o_1 is denoted by $o_1.A$. Exponents denote representations of different providers, i.e., o_1^2 is the representation of o_1 of provider 2.

4.1.1 Discrete Domain

For discrete domains, uncertain values are represented as probability distributions. Let X be the domain of attribute A and $a : X \rightarrow [0, 1]$ the probability distribution of A . The consistency of the A attribute two representations of object o is

$$c = \frac{1}{2} \left(\sum_{x \in X} o^1.a(x) \cdot \text{sgn}(o^2.a(x)) + \sum_{x \in X} o^2.a(x) \cdot \text{sgn}(o^1.a(x)) \right).$$

As outlined at the beginning of this chapter, for each provider, the probabilities for all values are summed up, for which the probability specified by the other provider is larger than 0 (accomplished using the sgn function).

Table 4.1 shows an example where two data providers offer different values for an uncertain enumeration-typed color attribute. The numbers in the table are the probabilities for the colors. The consistency of the values is 0.65.

	red	green	blue
provider 1	0.5	0.5	0
provider 2	0	0.8	0.2

Table 4.1: Example values

4.1.2 Continuous Domain with Pdf

For continuous domains, uncertain values are represented as probability density functions. Let X be the domain of attribute A and $a : X \rightarrow \mathbb{R}_0^+$ the probability density function of A . The consistency of the A attribute two representations of object o is

$$c = \frac{1}{2} \left(\int_{x \in X} o^1.a(x) \cdot \text{sgn}(o^2.a(x)) dx + \int_{x \in X} o^2.a(x) \cdot \text{sgn}(o^1.a(x)) dx \right).$$

In Figure 4.1, two providers specify different probability density functions for a temperature. In this example, the consistency is 0.7.

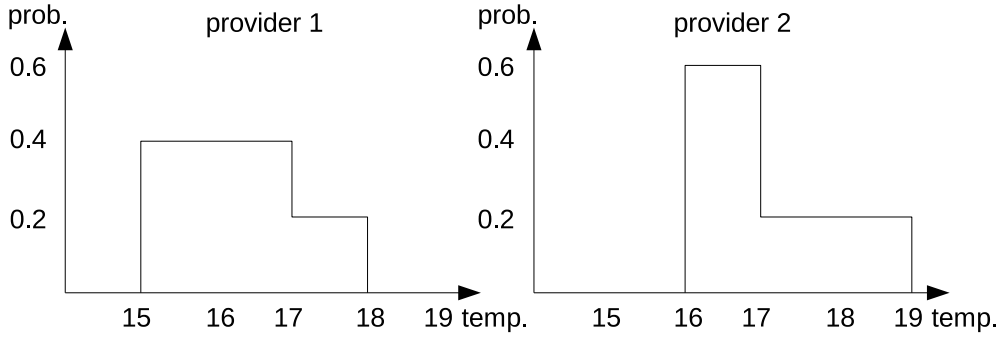


Figure 4.1: Temperature specified by probability density functions

4.1.3 Continuous Domain without Pdf

When positions are given as probability density functions, the definition from Section 4.1.2 can be applied. In this section, we present a measure for inconsistency where positions are represented by sets of areas with probabilities without making assumptions about the distribution inside the areas (cf. Chapter 3). As a consequence, the consistency model does a kind of best case / worst case estimation and generates a pair of consistency values ($c = (c_l, c_u)$).

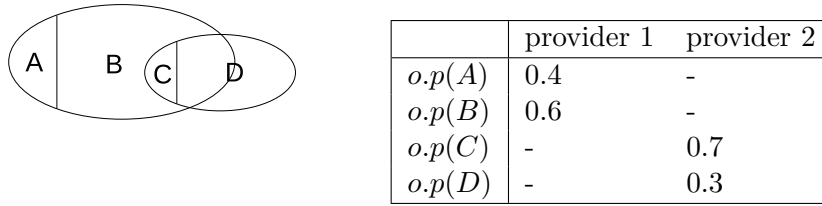


Figure 4.2: Uncertain areas and corresponding probabilities

In Figure 4.2, two providers offer uncertain position information for the object o . The uncertain position is represented by areas and a probability distribution, i.e., $o.p(A)$ is the probability that the position of object o is inside area A . For the probabilities declared as “-”, the provider does not explicitly specify a value, for calculating the consistency, we treat them as 0.

For calculating the lower bound of the consistency value, for each provider the probabilities of all areas that are contained in an area specified by the other provider, are summed up. The two values are added and normalized to the interval $[0, 1]$.

$$c_l = \frac{1}{2} \left(\sum_{X|\exists Y: X \subseteq Y \wedge o^2.p(Y) > 0} o^1.p(X) + \sum_{X|\exists Y: X \subseteq Y \wedge o^1.p(Y) > 0} o^2.p(X) \right)$$

4 Consistency Model

The upper bound is calculated similarly by including all overlapping areas.

$$c_u = \frac{1}{2} \left(\sum_{X|\exists Y: X \cap Y \neq \emptyset \wedge o^2.p(Y) > 0} o^1.p(X) + \sum_{X|\exists Y: X \cap Y \neq \emptyset \wedge o^1.p(Y) > 0} o^2.p(X) \right)$$

For the example in Figure 4.2, the consistency is (0.35, 0.8). Table 4.2 shows some more examples for inconsistent position information and resulting consistency values.

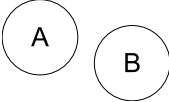
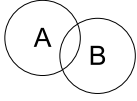
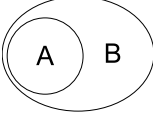
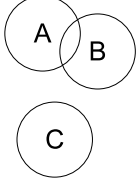
	attribute	provider 1	provider 2	consistency
	$o.p(A)$	0.4	0.3	(1,1)
	$o.p(B)$	0.6	0.7	
	$o.p(A)$	1	-	(0,0)
	$o.p(B)$	-	1	
	$o.p(A)$	0.4	0.4	(1,1)
	$o.p(B)$	0.6	0.6	
	$o.p(A)$	1	-	(0,1)
	$o.p(B)$	-	1	
	$o.p(A)$	1	-	(0.5,1)
	$o.p(B)$	-	1	
	$o.p(A)$	0.4	-	(0.65,1)
	$o.p(B)$	-	0.3	
	$o.p(C)$	0.6	0.7	

Table 4.2: Inconsistent position information and resulting consistency values

4.1.4 3D-Domain

For the evaluation of inconsistencies in multiply represented 3D building models from different context providers, we proposed the approach described in [Pet09], which will be illustrated in the following.

The approach aims at the evaluation of inconsistencies between an input model and a reference model. For every face in the usually higher detailed and accurate reference model, a local coordinate system is constructed. In the case of horizontal faces, this is the face's normal vector and its cross product with the x-axis of the model coordinate system, complemented to a right-hand-system. For all other faces,

4.1 Inconsistency on the Context Information Layer

the z-axis is used instead of the x-axis. Input model faces relevant for the comparison to the current reference model face are compiled according to their type, where a distinction between wall and roof faces is made. This set of faces is further downsized by comparing the normal vectors. However, instead of using an angular threshold, only faces with opposite direction to the reference face are removed as these are not likely to represent a building feature similar in both models. The relevant faces are then projected into the local coordinate system and the intersection of the current reference face and the projected relevant face is computed. If an intersection polygon exists, its area is computed. However, faces exceeding a distance threshold with their mean distance to the reference face are excluded. This is necessary, as for the final consistency value distance and angular inconsistencies will be merged with the areal differences. Faces exceeding the distance threshold are nevertheless regarded in the consistency computation by their missing area.

From the three characteristic values—distance, angle and intersection area—the consistency value per face is computed as

$$c = \frac{1}{A} \sum_i \left(1 - \frac{d_i}{d_{\max}} - |\alpha_i| \right) \cdot A_i$$

with d_i being the mean distance and α_i the mean angle between face and the reference face, A_i being the area of the respective input face and A the area of the reference face. The resulting value in the interval $[0, 1]$ may for example be used to color the input face (see Figure 4.3).

In order to test the approach, differently detailed data from four sources was used. Ordered by descending level of detail and accuracy, these sources are: Hand-crafted models of landmarks from a city model derived from terrestrial data collection; medium detailed models from airborne photogrammetric data collection; building models from the same city model, however simplified using the approach by [Kad07]; OpenStreetMap ground plans extruded to 3D block models.

Figure 4.3 depicts the results for the Rosenstein museum models. In the OpenStreetMap model, the bigger differences in the longer walls in contrast to medium inconsistencies in the shorter sides reproduce quite well the shift of the complete building model. In the generalized model, the strongly simplified roof structure shows the most distinct inconsistency to the reference model, with slight differences for the atrium and flat roof sides. The city model from airborne data collection, however, holds high consistency in the main wall planes. As both of these models are provided by the city surveying office, this is most likely due to the shared data base and accurately measured ground plans. The slight inconsistencies in the roof planes stem mainly from differently modeled roof angles, whereas the atrium without a match in the model from airborne data acquisition is marked clearly visible.

While the fusion of differently detailed 3D building models is a very difficult task, the minimization of some of the detected inconsistencies is feasible. One example are ground plan inconsistencies, which may evolve from low quality sensor data (e.g. OpenStreetMap models) or averaging operations during the generalization process.

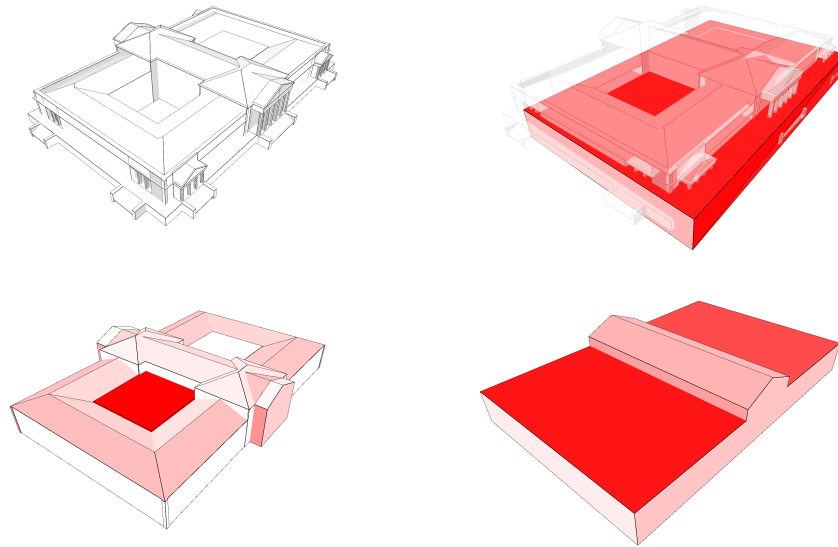


Figure 4.3: Clockwise: Inconsistencies of OpenStreetMap model, generalized model and city model from airborne data collection in comparison to the city model from terrestrial data collection (upper left)

To allow for the adjustment of an input model to the ground plan of a given reference model, first, the reference model is analyzed and approximating planes similar to those described in the generalization approach by [Kad07] are computed. However, using the faces' areas as weights, these are constructed as the planes with maximum weight for a set of parallel faces below a given distance threshold. These planes are further classified according to their adjacency to the ground plan.

The same step is done for the input model, where planes that are connected to the ground plan are considered moveable. Each of these moveable planes is then adjusted to the reference plane with the best ratio between reference plane weight and distance between both planes. In order to adjust the complete 3D building structure, the remaining non-vertical approximating planes are set in relation to the aforementioned planes using distance ratios that only consider the x- and y-coordinates of the intersection points. To avoid topological errors, the slope of the roof planes has to change during the fitting, which is implemented by strictly maintaining the height levels of every point.

4.2 Situation Recognition

The Nexus system is designed as an open system where any commercial and non-commercial provider can “place” context models and situation templates into the system. A consequence of this could be that several different templates for the recognition of the same situation may be available in the repository. While this is

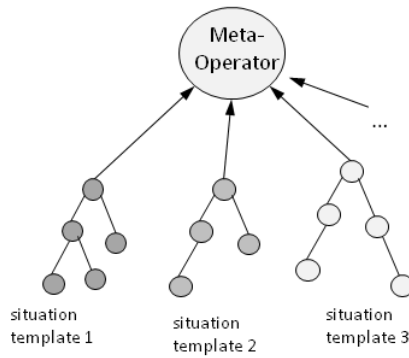


Figure 4.4: Structure of a meta-template

desirable on the one hand, because thus a more robust and reliable detection is possible. But on the other hand, non-trivial problems could arise e.g. if several templates $T_S = \{t_1, t_2, \dots, t_n\}$ should actually detect the same situation S , but the templates are contradictory in their results. To solve this problem we evaluated and adapted the approach of [HAES08] by expanding our template concept (introduced in Chapter 3.4.2) to an additional new class of templates—the so called *meta-template*. This meta-template combines exactly these templates in one operator (the so called *meta-operator*) which (should) recognize all the same situation-type S (see figure 4.4). One of the biggest advantages of this method is the exploitation of the existing template-structure and the implicit integration of consistency in the already existing confidence value. Another advantage is the automatic adaption of the confidence-metric of each single template $t \in T_S$ using an Reward-and-Punishment approach for refinement of the metric.

Reward-and-Punishment method We differ, accordingly the size of the feedback signal, between several *set of events*. Notice that the method can be adjusted due to the size of the feedback signal by adjust the number of *sets*. Due to our boolean feedback signal we define two sets of events $S \in \{O, N\}$, where O includes all templates which detect "situation occurs" and N includes all templates which detects "situation does not occur". Thus we define:

$$\begin{aligned} O &= \{t_i, \dots, t_j\} \in S \\ N &= \{t_k, \dots, t_l\} \in \neg S \end{aligned}$$

The next fundamental step is to assign each template to one of the *set of events*. So we determine first the uncertainty-values (probability P and confidence C) of each template $t \in T_S$ as mentioned in chapter 3.4.2 using a simple message-passing-algorithm. In the next step each template will be assigned to the corresponding *event-set* according to a classification function $c(t)$ using a *threshold* and the previ-

4 Consistency Model

ously determined probability-value $P(t)$.

$$c(t) = \begin{cases} t \Rightarrow O & , \text{ if } P(t) > \text{threshold} \\ t \Rightarrow N & , \text{ else.} \end{cases}$$

Subsequently, a scoring-function for all event-sets is determined as

$$\text{score}(\text{set}) = \frac{\sum_{t \in \text{set}} P^{1+\sigma C(t)}(t)}{M}.$$

Where $\text{set} \in \{O, N\}$ and σ represents a weighting factor and M stands for a normalization factor.

In the following we determine the set with the highest score—hereafter named as set_{won} . According to this result the templates $t \in \text{set}_{\text{won}}$ are rewarded and the templates in the other sets are punished.

The reward- and punishment-value for each template is calculated by a simple Δ -Rule, determined by the formula:

$$\Delta = |\text{score}(\text{set}_{\text{won}}) - \max\{\text{score}(\text{set}_{\text{lose}1}), \dots, \text{score}(\text{set}_{\text{lose}N})\}|$$

According to this the new confidence-value of each single template t is adapted by

$$C_{\text{new}}(t) = \begin{cases} C_{\text{old}}(t) + \Delta & , \text{ if } t \in \text{set}_{\text{won}} \\ C_{\text{old}}(t) - \Delta & , \text{ else.} \end{cases}$$

As already mentioned above the confidence-value C and probability-value P of the whole situation recognition process (i.e. the confidence and probability of the meta-template $C(\text{meta})$) is calculated by averaging all single confidence- and probabilities-values of the templates $t \in \text{set}_{\text{won}}$ as follows:

$$C(\text{meta}) = \frac{\sum_{t \in \text{set}_{\text{won}}} C(t)}{|\text{set}_{\text{won}}|}$$

...and analogously:

$$P(\text{meta}) = \frac{\sum_{t \in \text{set}_{\text{won}}} P(t)}{|\text{set}_{\text{won}}|}$$

5 Trust Model

5.1 Introduction

In context-aware systems users need and highly depend on information, services and applications provided by various service providers and other users. Unfortunately, it is seldom possible to verify on our own whether the information received is correct, whether a service is reliable or whether applications will be useful and run stable. Instead, we often have to rely on the experiences and expertise of others.

With a *reputation system* users can share their knowledge and opinions about other users. The reputation system collects and systematically evaluates the opinions of all users about the trustworthiness of others. On request it computes the resulting trust value for the requested entity according to a *trust model*.

The use of reputation systems has been proposed for various applications, for example to rate products and product reviews, to validate the trustworthiness of sellers and buyers in online auctions (e.g., in eBay) and to detect free-riders in peer-to-peer networks.

However, by relying on recommendations from others we take a certain risk. Although some of the recommendations might be valuable to us, others might be useless or even misleading and harmful because some recommenders might have malicious intentions or not the required competence. Thus, we have to find out and to decide carefully whom we can trust. Unfortunately, we will not always be able to validate the trustworthiness of everyone providing recommendations on our own either, and we might want to look at recommendations about the trustworthiness of the recommenders as well, and so on. Finally, we end up with a complex graph of trust relations. In order to evaluate the trust graph we have to know which conclusions we can draw from the statements in the trust graph and how we can compute the resulting strength of a derived trust relation.

Moreover, the authenticity of all exchanged opinion statements should be protected, e.g., with digital signatures, to prevent manipulations and to make the evaluation verifiable to the users. Digital signatures are only useful if the users can identify the signature key holder. If no global trusted public key infrastructure is available, users can share their knowledge about the ownership of keys in a so-called *web of trust* (e.g., the PGP/GnuPG *web of trust* [ACGW99]) by exchanging digitally signed identity certificates. However, these authenticity statements are only useful if the users can verify that the issuer is trustworthy to verify the ownership of public keys. Trust and authenticity evaluation are thus highly interdependent.

Therefore, this chapter proposes an integrated approach to evaluate uncertain and possibly conflicting trust and authenticity statements¹.

¹parts of this chapter have been published in [GHS08], [Gut08] and [Gut09]

In section 5.2 we present background information on trust and related work. In section 5.3 we propose a new model for trust and authentication statements and computation methods. In section 5.7 the new reputation system is finally evaluated and the advantages of the system in comparison to existing approaches are shown.

5.2 Related Work and Fundamentals

5.2.1 Trust, Trustworthiness and Reputation

Before discussing if and how trust can be modeled and formally represented it should be clarified what the term “trust” might mean in particular. There exists a nearly unmanageable field of definitions for the term “trust” in literature. Trust has, for example, been defined as

- “the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context.” [GS00]
- “a simplifying strategy that enables individuals to adapt to complex social environment, and thereby benefit from increased opportunities” [EC95, p. 38].
- “a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.” [Gam88]

In consideration of this pluralism, trust could be defined abstractly as a *multi-relational concept* only [GHS08, HS07, HS09]: We say that a *truster* trusts a *trustee* (e.g., a person, an institution or a technical system) in a certain *context*, if the truster has confidence in the *competence* and *intention* of the trustee and therefore believes that the trustee acts and behaves in an expected way, which does not harm the truster. We can therefore distinguish two categories of trust:

- *Competence trust*: Trust in the *capability* of a person, in an institution or in the functionality of a machine or a system.
- *Intentional trust*: Trust in the *moral integrity* (benevolence) of a person.

The *trust relation* between truster and trustee can be characterized as follows:

- *Symmetry*: Trust relations are *not symmetric* in general, i.e., if A trusts B, this does not necessarily imply that B trusts A. Thus, trust relations can be represented as unidirectional relations from the truster to the trustee.
- *Transitivity*: One can find contradictory opinions in literature about the question whether “A trusts B” and “B trusts C” implies “A trusts C”. According to our position trust is *not transitive*, because it is very well possible that A trusts B for performing certain actions, but not for giving recommendations².

²it is nevertheless possible to build trust chains under certain conditions (see subsection 5.2.4)

Therefore, the assumption of transitivity (e. g., in [HCD04, KR03]) can lead to counterintuitive effects.

- *Time Variance*: Trust may *change over time*, e. g., increase after successful co-operations and decrease after periods without interactions. This aspect will not be discussed further, though.

Trust is inherently related to risk and uncertainty. If everything would be predictable or perceivable, trust would not be required. The one who has confidence in someone or something often dares a possible harm: By acting someone exposes himself to a the risk of being disappointed in his expectations.

Some people claim that “real” trust starts there, where no probability estimation could be given because of the lack of historical-empirical data. For them, trust should make a risk calculation dispensable so as to reduce complexity. “Real” trust would thus become relevant where no probability estimations can be given. However, in the context of reputation systems the term “trust” is used to refer to a risk estimation which helps to decide whether or not to choose a risky action. Unfortunately, trust is often based on a limited amount of experience, incomplete knowledge and questionable assumptions. Therefore, one should be aware of the degree of uncertainty of trust values.

If a truster beliefs that he has not enough knowledge about the trustee or that he is not competent to decide on the trustworthiness of the trustee we talk about *ignorance*.

Often you act and you are not aware of the fact, that by acting you do at the same time trust in something or someone. Trust is often unconscious, is thus a way to reduce *complexity* [Luh79] since you are not forced to explicitly control a situation which would absorb mental capacities and therefore produce extra complexity. If someone is asked to think about his (possibly unconscious) trust in others, to verbalize and to explicitly express his opinion about the trustworthiness of some trustee (including a *trust value* as a quantification of the degree of trustworthiness) to others, we obtain a *trust statement*. Trust statements make it possible to exchange opinions with others. If someone is considered trustworthy for issuing truthful and valuable trust statements (recommendation), then his opinions can be used to broaden one’s own view, to learn from the experience of others and to come to more reliable trustworthiness estimations. Users may also exchange opinions about the trustworthiness of users for giving recommendations. This type of trust referring to the ability for giving trustworthy recommendations will be called *recommendation trust* in the following. To clarify the distinction we will call the direct, not recommending type of trust *functional trust*. Note that *trust category* and *trust type* are orthogonal classification dimensions as shown in Figure 5.1.

The idea of issuing and exchanging trust statements leads to the design of *reputation systems*: Information systems that automatically and systematically gather trust statements of different issuers, accumulate and amalgamate the different subjective opinions and trust values according to the trustworthiness of their issuers in order to compute a resulting estimation of the trustworthiness of a given trustee,

Type	Category	Intention, Benevolence	Competence
Functional Trust		Trustee wants to safely land the plane (is not a terrorist).	Trustee knows how to safely land a plane (is a skilled pilot).
Recommendation Trust		Trustee gives reliable recommendations about whether others want to safely land the plane (can discover terrorists).	Trustee gives reliable recommendations about whether others can safely land a plane (can recommend skilled pilots).

Figure 5.1: Exemplification of trust types and categories (trust context “landing a plane”)

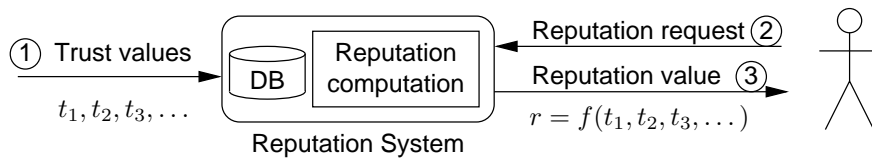


Figure 5.2: Reputation System

which may then serve as basis for decision making (see Figure 5.2). This resulting opinion contains (in contrast to the previous trust opinions) not only the opinion of one single individual, but a mixture of opinions of different individuals. To distinguish between these different types of opinions we will use the term *trust value* for the opinion of *one single entity* based only on own knowledge and experiences, whereas a *reputation value* represents a value computed from the *opinions of different entities*.

A reputation value computed by a reputation system may serve as a more reliable basis for taking decisions than the own trust value alone and can thus have an influence on the decisions taken. However, the fact that someone has a high reputation should not have a direct influence on trust values, because trust values represent the *own* opinion only without influence of the opinions of others. If a truster has only low trust in a trustee but the trustee turns out to have a high reputation, then we cannot expect (and it would not be advisable either) that the truster will somehow “increase his trust” in the trustee (how ever he would do that), but it can be advisable for the truster to engage in a risky interaction with the trustee due to the high reputation value. (If this interaction is successful, then the *own* positive experience may lead to an increase of trust, though.) Whether the truster actually does act according to the recommendation of the reputation system is not predictable, as he is not obligated to follow this recommendation. Instead, he is free to base his decision on any mixture of both, his own trust and the computed reputation value. Reputation systems therefore cannot *establish* trust between different partners of

interaction, but they can convey interactions by giving the partners a broader and more reliable basis to estimate the trustworthiness of each other.

5.2.2 Modeling Trust

In order to get reputation systems work, empirical facts and circumstances need to be numerically (or symbolically) represented, i. e., the strength of trust relations has to be quantified and measured by an associated trust value. Various different models to represent and to compute with trust have been proposed [GS00, Mar94, Mau96, Jøs97, ACGW99, HKL00, KSGM03, Dem04, JIB07, Koh07].

Besides (positive) “trust” there exist also propositions for expressing neutral or negative opinions. Although definitions have been proposed for “distrust”, “un-trust”, “mistrust”, the “lack of trust” and “ignorance” (e. g., by Marsh [MD05] and Grandison [GS00]), there is no clear consensus. One could distinguish the following forms of negative and neutral opinions: A truster *distrusts* (or *mistrusts*) a trustee if he believes that the trustee will not behave as expected, either due to a lack of competence or due to a malicious intention (e. g., if he believes that the trustee will seek to betray and actively work against him). A truster is said to have *no trust* in a trustee if he believes that it is neither justifiable to consider the trustee trustworthy nor to consider him distrustworthy (also called *absence of trust*).

Simple trust models represent a trust value by a single value, either on a *discrete scale*, e. g., by a Boolean value (“trust”, “no trust”) or by a more fine-grained scale as in PGP/GnuPG [ACGW99] (“untrustworthy”, “marginal trust” and “full trust”), or on a *continuous scale*, e. g., as proposed by Maurer [Mau96] (trust values in the range $[0, 1]$) or by Marsh [Mar94] (trust values in the range $[-1, 1]$).

Not all proposed trust models cover the full range of possible trust values. Some allow to express only *positive* trust values in the range between “no trust” (represented by 0) and “full trust” (represented by 1), whereas other offer the possibility to assign also “distrust” (represented by -1). However, the semantics of the trust values is sometimes different in the proposed models. Even though reasoning with distrust requires great care (an enemy of your enemy is not necessarily your friend), negative trust values may be useful especially in applications, in which the possible harm of unsuccessful interactions is high.

It is important to allow entities to express uncertainty about their trust opinions and to record this degree of uncertainty. Without this possibility the task of gathering trust opinions could cause so-called *response errors*, i. e., people who are prompted for their opinion about the trustworthiness of a subject but who do not have a reliable opinion about the trustee in question might give rather speculative answers. The degree of trustworthiness of the opinions should be taken into account in the reputation evaluation in order to avoid that valuable reliable opinions get outvoted by unreliable speculations.

Most approaches allow therefore to express ignorance (e. g., “I *can not decide* whether I can trust him”) or the degree of uncertainty (e. g., “I am *quite sure* that I can judge his trustworthiness correctly”) of a trust opinion, either by a discrete value

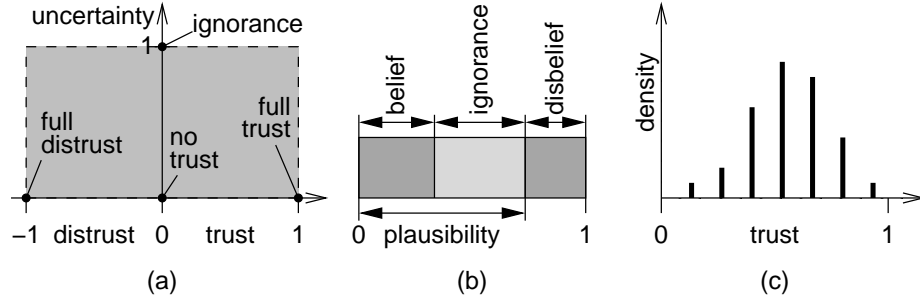


Figure 5.3: Different possibilities to represent trust values

(e. g., “don’t know” in PGP/GnuPG) or on a continuous, additional reliability scale. Trust values can be expressed for example by two independent continuous variables for trustworthiness and reliability, e. g., trust $t \in [-1, 1]$ and uncertainty $u \in [0, 1]$ (see Figure 5.3a), or by two continuous values with dependencies, e. g., Dempster-Shafer [Sha76] and related approaches [SF02] represent trust by an upper and a lower bound ($0 \leq \text{belief} \leq \text{plausibility} \leq 1$), which is equivalent to Jøsang’s opinion triangle [Jøs97] representing trust values by a belief (b), disbelief (d) and ignorance value (i) ($b, d, i \in [0, 1]$, $b + d + i = 1$, see Figure 5.3b). Credential Networks and related models proposed by Haenni [HKL00], Jonczy [JH05] and Kohlas [Koh07] also model trust values with belief, ignorance and disbelief values. However, trust values may contain either degrees of belief and ignorance, disbelief and ignorance, or belief and disbelief, but they cannot contain degrees of belief, disbelief and ignorance at the same time.

Furthermore, it is also possible to represent trust values as arbitrary discrete distribution functions [Gut07] (see Figure 5.3c).

An important yet difficult task is to define the semantics of the trust values to ensure the correct interpretation of the trust statements. This may include

- defining an order relation between trust values (e. g., is “full trust” higher than “marginal trust”?),
- specifying whether differences between trust values can be meaningfully compared (e. g., is the step between “untrustworthy” and “marginal trust” comparable to the step between “marginal trust” and “full trust”?),
- specifying whether the ration between two trust values is meaningful (e. g., is “0.9” twice as trustworthy as “0.45”?), and finally
- assuring that a certain trust value (e. g., “0.45”) means the same to all users (e. g., does the trust value represent a probability?).

The choice for an approach to represent reputation values may depend on the requirements and context of the application. However, approaches with the possibility

to represent uncertainty make it easier to avoid counterintuitive effects during the evaluation of trust relations.

If a trustor has no information about a certain trustee, it is reasonable to assign a trust value corresponding to “ignorance” as default value. If the trust model does not allow to represent ignorance, the lowest possible trust value is a safe choice to prevent malicious entities to get rid of bad reputation by changing their identity (“whitewashing”).

5.2.3 Classification of Reputation Systems

We can distinguish 3 basic types of reputation systems (see Figure 5.4) with different approaches to calculate reputation values:

- Type A: Flat reputation systems
- Type B: Recursively weighting reputation system
- Type C: Personalized reputation system with *trust anchor*

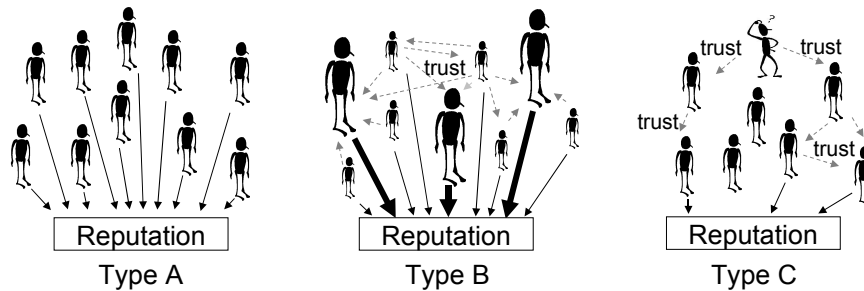


Figure 5.4: Classification of reputation systems

Type A reputation systems (e.g., in eBay) are very simple. The reputation values are computed from all (or random samples of all) available trust opinions of all entities. The opinion of each entity has the same weight, i.e., liars have the same influence on the resulting trust value as honest entities. Note that (without additional measures) the collected opinions will normally not be *representative* for the group of users because the users themselves decide whether or not they want to “participate” in the “survey”, i.e., to publish trust statements or not. This is especially critical if a single person can create a high number of (apparently different) entities or user accounts in a reputation system. In that case a single person can outvote all other entities by a so-called *Sybil-attack*.

Type B reputation systems (e.g., the Basic EigenTrust algorithm [KSGM03]) try to improve the quality of the computed reputation value by increasing the weight of higher ranked opinions. Reputation values of all entities can therefore be computed iteratively: The new reputation values of all entities are computed from the opinions of all other entities weighted by their reputation values of the last iteration. However,

the group of participating users is still not representative, and it is still possible that a large group of colluding malicious entities dominates the “public opinion” and manipulates the computed reputation values.

Type C reputation systems (e. g., as proposed by Maurer [Mau96], Jøsang [JI02] and Gutscher [Gut07]) aim to resist this kind of attacks. They always start with a “safe” set of *a priori* trusted entities (the so-called *trust anchor* or *trust root*), which normally consists of the requester himself. First, only the opinions of the *a priori* trusted entities are taken into account. Next, also the opinions of entities which have been found to be trustworthy in the previous iteration are taken into account, too. This process is repeated until the opinions of all “reachable” trustworthy entities are included in the reputation value computation. Note that opinions of untrustworthy entities are ignored as long as the opinions of the trust anchor entities are correct. In contrast to the previous reputation systems, Type C reputation systems are *personalized*, because requesters with different trust anchors will in general obtain different reputation values for the same trustee. In the following, we will focus on Type C as the most advanced type.

5.2.4 Reasoning with Trust Relations

Once the attributes, properties and the quantitative representation of trust values have been agreed upon, the process of evaluating trust relations has to be defined. For this purpose, trust models (explicitly or implicitly) define a set of inference rules, which define whether and which conclusions (new reputation relations) one can draw from a set of given trust relations. Inference rules define the made assumptions on the *transitivity property* of trust relations, but also prerequisites and restrictions depending on the type and attributes of the involved trust relations as well as on the associated trust values.

Most trust models assume that trust is *not transitive* in general, but Instead, they differentiate between functional and recommendation trust and define via inference rules which trust relations can be combined to trust chains. The trust model proposed in [Gut07] for example allows to specify for each recommendation trust relation a limit h for the allowed remaining length of trust chains (*recommendation hops*). A recommendation trust relation with $h = 1$ expresses the belief of the truster that entities recommended by the trustee are trustworthy in the sense of functional trust, whereas recommendation trust relations with $h = 2$ expresses the belief of the truster that entities recommended by other entities recommended by the trustee are trustworthy in the sense of functional trust, etc. The following trust derivation rules define how trust chains can be constructed³:

1. Recommendation trust from A to B with $h_1 = 1$ can be combined with functional trust from B to C to a new functional trust relation from A to C.

³it is assumed that all involved trust relations refer to the same trust context

2. Recommendation trust from A to B with $h_1 = n + 1$ can be combined with recommendation trust from B to C with $h_2 = n$ to a new recommendation trust relation from A to C with $h = n$ (for $n \geq 1$).

These rules would allow to combine trust relations only if the number of recommendation hops matches *exactly*, which could be seen as an counterintuitive and thus inappropriate restriction. Therefore, it is possible to make the additional assumption that recommendation trust with a limit of $h = n + 1$ implies recommendation trust with a limit of $h = n$ recommendation hops (for $n \geq 1$).

5.2.5 Computation of Reputation Values

Once new reputation relations have been derived an associated reputation value has to be computed. The computation of reputation values in Type A reputation system is very simple, e. g., the arithmetic mean all trust values is a reasonable choice. Reputation computation in Type B can be done iteratively. First, initial reputation values are computed as in Type A reputation systems. Then, new reputation values for all entities are calculated from the opinions of all entities weighted by their associated reputation values of the last iteration. This process is repeated and the reputation values has converged.

In Type C reputation systems the reputation evaluation process starts from the trust anchor specified by the requester: First, a set of all trust relations issued by *a priori* trusted entities is compiled. Then, the trust inference rules are applied to the relations in this set and all derivable trust relations are added to this set. The last step is repeated until all inferable trust relations are already contained in the set. Next, the trust values for the derived statements can be computed. Here, we distinguish two different classes of reputation value computation approaches, an *operator-based* and a *probability-theoretical* approach: In the *operator-based* approach the initial opinions are deterministic, but the inference rules are probabilistic. In the *probability-theoretical* approach the initial opinions are probabilistic, but the inference rules are deterministic.

5.2.5.1 Operator-based Approach

The trust value of the new trust relations is computed by successively combining all relevant parallel or concatenated trust relations to one single resulting trust relation (see Figure 5.5). First, all opinions that are not relevant to derive the requested statement are removed. Next, in each step, two parallel or concatenated trust relations are replaced by one resulting trust relation. The trust value of the new relation is computed from the two trust values of the replaced trust relations by a trust combination *operator*. This process is repeated until we reach a graph with one resulting trust relation from the requester to the final trustee.

A simple example with operators from probability theory is shown in Figure 5.5: Trust values are represented by values in the range $t \in [0, 1]$ (thus, no uncertainty can be expressed). The resulting reputation value for a concatenation of two trust

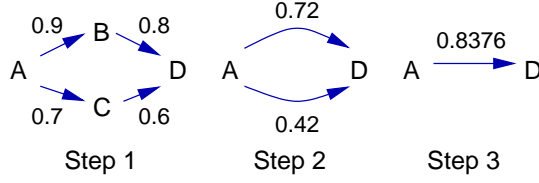


Figure 5.5: Operator-based reputation computation

relations is $t = t_1 t_2$, the resulting reputation value for parallel relations is $t = 1 - (1 - t_1)(1 - t_2) = t_1 + t_2 - t_1 t_2$.

Corresponding operators for the *belief/ignorance/disbelief* confidence value representation ($t = (b, i, d)$) have been proposed for example by Dempster-Shafer [Sha76] and related approaches [SF02] as well as Jøsang [Jøs97].

The following *conjunction, disjunction* and *negation operators* have been proposed by Baldwin [Bal87] and Jøsang [Jøs97]:

$$\begin{aligned}
 t_x \wedge t_y &= \begin{pmatrix} b_x b_y \\ i_x i_y + i_x b_y + b_x i_y \\ d_x + d_y - d_x d_y \end{pmatrix} \\
 t_x \vee t_y &= \begin{pmatrix} b_x + b_y - b_x b_y \\ i_x i_y + i_x d_y + d_x i_y \\ d_x d_y \end{pmatrix} \\
 \neg t_x &= \begin{pmatrix} d_x \\ i_x \\ b_x \end{pmatrix}
 \end{aligned}$$

The *belief* value b of a conjunction can be interpreted as the probability that both input values are *belief*, d as the probability that at least one input value is *disbelief*. The remaining probability mass is assigned to *ignorance*. The disjunction operator is constructed accordingly. The negation operator swaps the *belief* and *disbelief* values. From these operators we can derive the corresponding truth tables for the *discrete* trust values *belief, ignorance* and *disbelief* (see Figure 5.6).

\wedge	+	\emptyset	-
+	+	\emptyset	-
\emptyset	\emptyset	\emptyset	-
-	-	-	-

\vee	+	\emptyset	-
+	+	+	+
\emptyset	+	\emptyset	\emptyset
-	+	\emptyset	-

\neg	
+	-
\emptyset	\emptyset
-	+

Figure 5.6: Deterministic conjunction, disjunction and negation operators (Baldwin, Jøsang)

A *recommendation operator* (\otimes) concatenates two trust relations i. e., it combines a trust relation from an entity E_A to an entity E_B with trust value t_x with a trust

relation from E_B to an entity E_C with trust value t_y to one single trust relation from E_A to E_C with trust value $t_x \otimes t_y$ (see Figure 5.7).

Jøsang's recommendation operator [Jøs97] follows the advice of trusted recommenders and ignores unknown and distrusted recommenders (*ignorance favoring strategy*). The operator and the corresponding truth table are shown in Figure 5.7.

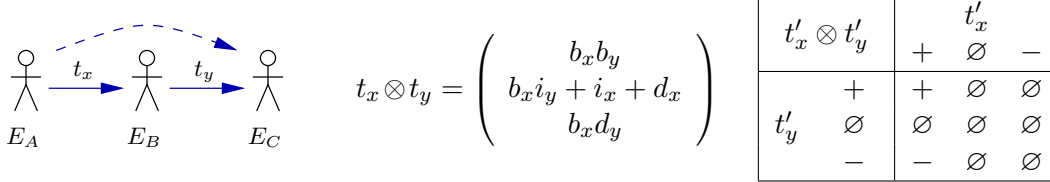


Figure 5.7: Recommendation operator (Jøsang)

A *consensus operator* (\oplus) combines the trust values of two trust relations that refer to the same proposition. Jøsang [Jøs97], Dempster-Shafer [Sha76] and Yager [Yag87] have proposed the consensus operators shown in Figure 5.8 (undefined values are indicated by \diamond). Intuitively, the combination with *ignorance* does not change discrete

$$t_x \oplus t_y = \frac{1}{i_x + i_y - i_x i_y} \begin{pmatrix} b_x i_y + i_x b_y & & \\ i_x i_y & & \\ d_x i_y + i_x d_y & & \end{pmatrix}$$

\oplus	+	\emptyset	-
+	\diamond	+	\diamond
\emptyset	+	\emptyset	-
-	\diamond	-	\diamond

$$t_x \oplus t_y = \frac{1}{1 - b_x d_y - d_x b_y} \begin{pmatrix} b_x b_y + b_x i_y + i_x b_y & & \\ i_x i_y & & \\ d_x d_y + d_x i_y + i_x d_y & & \end{pmatrix}$$

\oplus	+	\emptyset	-
+	+	+	\diamond
\emptyset	+	\emptyset	-
-	\diamond	-	-

$$t_x \oplus t_y = \begin{pmatrix} b_x b_y + b_x i_y + i_x b_y & & \\ i_x i_y + b_x d_y + d_x b_y & & \\ d_x d_y + d_x i_y + i_x d_y & & \end{pmatrix}$$

\oplus	+	\emptyset	-
+	+	+	\emptyset
\emptyset	+	\emptyset	-
-	\emptyset	-	-

Figure 5.8: Jøsang's (top), Dempster-Shafer's (middle) and Yager's (bottom) consensus operators

trust values, and in the cases of Dempster-Shafer and Yager the combination of two identical discrete trust values t' results in t' . The operators differ in their conflict handling strategy. Dempster-Shafer's operator is defined only for $1 - b_x d_y - d_x b_y > 0$, i. e., it is undefined for the combinations of *belief* with *disbelief*. The probability mass of undefined combinations is eliminated and b , i and d are re-normalized so that $b + i + d = 1$. Jøsang's operator is defined only for $i_x + i_y - i_x i_y > 0$, i. e., it is, in addition, undefined for the combinations of two belief values and of two disbelief values, which is counterintuitive as the trust values are identical. Jøsang, too, performs a re-normalization. These re-normalizations have the effect of completely

ignoring conflict and can thus lead to counterintuitive effects [Zad84]. Yager’s consensus operator [Yag87] assigns the probability mass of conflicting combinations to *ignorance*. This avoids the counterintuitive effects of re-normalizations, but conflict is then indistinguishable from ignorance. In security-critical applications, it can be very important to distinguish these cases and treat them differently. A high degree of conflict indicates that the trustee misbehaved in the past or that some recommenders are lying, whereas ignorance indicates merely a lack of information.

Reasoning with distrust requires great care to avoid possibly misleading conclusions. The following constellation is an example for a situation in which it is not obvious to decide which outcome should be considered the most “reasonable”: An entity B issues a (positive or negative) trust statement about C. Entity A wants to find out whether C is trustworthy, although A distrusts B. It would be possible to *ignore* statements from untrustworthy entities or to assume the *opposite* trust value. In the first case, A might lose possibly useful information, but the strategy is a safe choice. The second strategy is logically not sound (an enemy of your enemy is not necessarily your friend) and might produce misleading results, especially if B is aware of A’s strategy.

The operator-based approach has a major drawback which renders it mostly unemployable: The successive combination of trust relations is only possible if either the operators are distributive (which is not true for all non-trivial operators) or if the graph of trust relations has a special structure, i. e., if it is a so-called *directed series-parallel graph* (which is unlikely to happen). A simple example of a graph that can not be evaluated with operator-based approaches is shown in Figure 5.9.

The proposed workaround [JGK06] to leave out opinions (without any compensation) is not an acceptable solution, because this can significantly change the result.

5.2.5.2 Probability Theoretical Approach

In the *probability-theoretical* approach the trust values are interpreted as estimations for probabilities that the initial opinions have certain discrete confidence values (e. g., full belief, full disbelief, full ignorance). Thus, in contrast to the operator based approach, the initial opinions are now considered uncertain, whereas the the inference rules to draw conclusions are considered deterministic.

The computation of the trust value of a requested statement is therefore based on the evaluation of a random experiment. In the approach proposed by Maurer [Mau96] and Gutscher [Gut07] for example it is assumed that trust values are expressed by a trust value $t \in [0, 1]$, which is interpreted as the probability that the trust relation is *valid*. The resulting reputation value is the computed probability that the *requested* trust relation is *valid*, i. e., that it is possible to derive the requested trust relation from an initial *starting set*, which consists of all initially valid relations. For n initial trust relations (which can each be valid or invalid) there exist 2^n different possible starting sets. For each scenario the inference rules are applied and it is evaluated whether the desired reputation relation can be derived from the relations in the starting set. In each *successful* scenario we calculate the probability

that this scenario will occur from the trust values of the initial trust relations. The resulting reputation value is the sum of the calculated probabilities of all successful scenarios.

Example We consider the example shown in Figure 5.9. The trust relations b and e represent functional trust, the trust relations c and d recommendation trust with a limit of one hop and the trust relation a recommendation trust with a limit of two hops. The corresponding trust values are $t_a, t_b, t_c, t_d, t_e \in [0, 1]$. We can find three possibilities (*trust paths*) to derive a functional reputation relation from A to D: (a, b) , (d, e) or (a, c, e) . The table in Figure 5.9 shows for each possible starting set whether it is possible to derive a reputation relation from A to D as well as the probability of each successful scenario. The resulting reputation value r is the sum of the probabilities of all successful combinations: $r = (1 - t_a)(1 - t_b)(1 - t_c)t_d t_e + \dots + t_a t_b t_c t_d (1 - t_e) + t_a t_b t_c t_d t_e$.

This approach can be used to evaluate arbitrary trust graphs and thus avoids the problem mentioned in subsection 5.2.5.1, but usually has a higher computational complexity⁴. Similar evaluation algorithms can be applied if the trust model supports the expression of uncertainty or if trust values are expressed by probability distributions [Gut07].

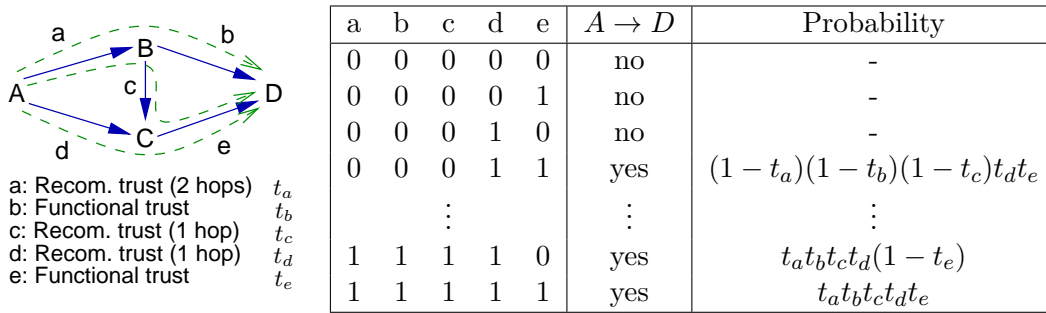


Figure 5.9: Probability theoretical reputation computation

5.2.5.3 Conflicting Opinions

If users can express both positive (supporting) and negative (refuting) opinions, then the combination of contradictory opinions of different users can lead to *conflicts* (in both operator-based and probability theoretical approaches).

The the degree of conflict reflects how strong the issuers of the initial opinions disagree with each other. The occurrence of conflict does not constitute a problem by itself, but it must be processed adequately.

In Credential Networks and Subjective Logic the probability mass associated with conflicting combinations is eliminated and the remaining probability mass is re-normalized. Zadeh [Zad84] has shown that conflict elimination and re-normalization

⁴note that there exist more efficient algorithms and approximations, see section 5.6

approaches (like Dempster’s rule of combination [Sha76]) can produce counter-intuitive effects.

In the following we therefore propose a new integrated approach to evaluate uncertain and conflicting trust and authenticity statements without eliminating conflict. This avoids the counter-intuitive effects of re-normalizations.

5.3 Model of Trust and Authenticity Statements

Due to the named drawbacks of other trust models we propose a new model to represent trust and authenticity statements, trust values, inference rules and a new approach to evaluate arbitrary networks of trust and authenticity opinions.

An *opinion* refers to a trust or authenticity statement H_j with an associated trust value t_j . A *first-hand* opinion is an opinion that is based only on the experience and knowledge of a *single* entity (the *trustor* or *issuer*) and that is *independent* of other opinions. A *second-hand* opinion is an opinion that is derived from other opinions and that is thus not independent.

We define *trust* as “a *unidirectional relation between a trustor and a trustee expressing the strong belief of the trustor that the trustee will behave as expected with respect to a particular capability within a particular context*” [Gut07]. Therefore we represent the standard form of a trust statement as follows:

$$\text{Trust}(\text{trustor}, \text{trustee}, r, h_{\min}..h_{\max}) \quad (5.1)$$

The *trustor* can be an entity or a key, the *trustee* an entity, a description or a key (see Table 5.1). An *entity* (E_A, E_B, \dots) can be a person, an organization, a network node, etc. referred to by a *local* identifier. To exchange opinions with others users have to use unique *descriptions* or *public keys* to refer to other entities. A *description* (D_A, D_B, \dots) consists of a list of names, identifiers or attributes that uniquely identifies the described entity. Entities may have several different descriptions. A *public key* (K_A, K_B, \dots) is the public part of an asymmetric key pair. The holder uses the key pair to sign trust or authenticity statements (certificates). An entity can use several different key pairs at the same time.

	Trust statements	
	Standard form	Internal form
Relation	$\text{Trust}(E_A, E_B, r, h_{\min}..h_{\max})$	$\text{Trust}(E_A, E_B, r, h, l)$
	$\text{Trust}(E_A, K_B, r, h_{\min}..h_{\max})$	$\text{Trust}(E_A, K_B, r, h, l)$
	$\text{Trust}(E_A, D_B, r, h_{\min}..h_{\max})$	$\text{Trust}(E_A, D_B, r, h, l)$
Certificate	$\text{Trust}(K_A, K_B, r, h_{\min}..h_{\max})$	$\text{Trust}(K_A, K_B, r, h, l)$
	$\text{Trust}(K_A, D_B, r, h_{\min}..h_{\max})$	$\text{Trust}(K_A, D_B, r, h, l)$

Table 5.1: Trust statements (relations and certificates)

The capability r refers to an application specific capability (r_1, r_2, \dots) or to the capability r_{PKI} , which represents the capability to honestly and carefully verify that a description uniquely refers to the holder of a particular key pair.

We distinguish different types of trust identified by a different number of recommendation hops (h): *Functional trust* expresses the belief that the trustee *has* the capability r and is described by $h = 0$. *Recommendation trust* for $h = 1$ hop expresses the belief that the trustee can *recommend* someone with capability r , *recommendation trust* for $h = 2$ hops that the trustee can *recommend someone who can recommend* someone with capability r , etc. Each standard form trust statement can specify the desired range of recommendation hops $h_{\min}..h_{\max}$.

For the evaluation of trust statements we need in addition trust statements in the slightly different *internal form*. These trust statements refer not to a range, but to a single recommendation hop value $h \geq 0$ and they have an additional parameter, the *chain length* $l \geq 1$:

$$\text{Trust}(\text{trustor}, \text{trustee}, r, h, l) \quad (5.2)$$

Trust is not transitive in general, but trust statements can be combined in certain cases to trust chains according to the transitive trust inference rule Equation 5.9 described in subsection 5.5.1. The chain length l of the derived trust statement refers to the number of first-hand trust statements in the trust chain.

Authenticity statements express the strong belief of the issuer that a description belongs to an entity, that a public key belongs to an entity or that a description belongs to the holder of a public key:

$$\text{Auth}(\text{issuer}, \text{actor}_1, \text{actor}_2) \quad (5.3)$$

The *issuer* is an entity or a public key, *actor*₁ and *actor*₂ are entities, descriptions or public keys. All four possible combinations are listed in Table 5.2⁵.

	Authenticity statements
Relation	$\text{Auth}(E_A, K_B, E_B)$
	$\text{Auth}(E_A, D_B, E_B)$
	$\text{Auth}(E_A, K_B, D_B)$
Certificate	$\text{Auth}(K_A, K_B, D_B)$

Table 5.2: Authenticity statements (relations and certificates)

5.4 Trust Values

This section introduces discrete and continuous trust values as well as operators for reasoning with discrete trust values. Users express their opinions with continuous

⁵certificates can not contain local identifiers for entities (E_A, E_B, \dots) because they would be meaningless to other entities

trust values while the discrete trust values are used internally only for reasoning with opinions.

5.4.1 Representation of Discrete and Continuous Trust Values

Users can have different and possibly conflicting opinions about trust and authenticity statements. Therefore, we can not definitively decide whether a statement H is “true” or “false”. We can only evaluate known indications that *support* or *refute* H . It is possible that neither supporting nor refuting or that both supporting and refuting indications for H are found. Therefore we describe knowledge of supporting and refuting indications *independently*. For each statement H we introduce the *propositions* H^+ and H^- to describe that the reputation system is aware of indications that imply that H must be true and that H must be false, respectively. We also introduce the four *discrete* trust values *belief* (+), *ignorance* (\emptyset), *disbelief* (–) and *conflict* (\pm) to represent the four possible combinations of these propositions (see Table 5.3). They can be seen as “truth values” of a paraconsistent logic [Gut08].

Propositions	Discrete trust value	Semantics
$\{H^+\}$	$t' = +$ (<i>belief</i>)	“the indications imply that H must be true”
$\{\}$	$t' = \emptyset$ (<i>ignorance</i>)	“there are no relevant indications about H ”
$\{H^-\}$	$t' = -$ (<i>disbelief</i>)	“the indications imply that H must be false”
$\{H^+, H^-\}$	$t' = \pm$ (<i>conflict</i>)	“the indications imply that H must be true and that H must be false at the same time”

Table 5.3: Discrete trust values

As statements can in fact not be both true and false at the same time we can conclude that *first-hand* opinions can not have the trust value *conflict*. However, if we combine statements of *different* (disagreeing) entities, it is possible to find both H^+ and H^- , i. e., the trust value of derived (*second-hand*) opinions can be *conflict*. Conflict must not be confused with partial support and partial refutation (*ambivalent opinions*). An entity that has for example experienced some positive and some negative interactions can express this opinion with *continuous* trust values.

Continuous trust values $t = (b, i, d, c)$ with $b, i, d, c \in [0, 1]$ and $b + i + d + c = 1$ express *degrees* of belief, ignorance, disbelief and conflict. The value b represents the issuer’s subjective estimation of the probability that there are indications supporting (but no refuting) H . Similarly, d represents the subjective estimation of the probability that there are indications refuting (but no supporting) H . c represents the subjective estimation of the probability that there are both supporting and refuting indications for H at the same time, and i represents the subjective estimation of

the probability that there are neither supporting nor refuting indications for H . For the same reason as before, c must be zero in all first-hand opinions, whereas second-hand opinions can contain conflict. Nevertheless, ambivalent first-hand opinions can be expressed by continuous trust values with both $b > 0$ and $d > 0$. A user that has made many positive and few negative experiences can choose, for example, a first-hand trust value with $b = 0.7$ and $d = 0.1$ (i. e., $t = (0.7, 0.2, 0.1, 0)$). Thus, in first-hand statements b can be seen as the lower bound and $1 - d$ as the upper bound for the estimated subjective probability that H must be true.

The degrees of ignorance and conflict in resulting trust values have different meanings, and applications should handle high degrees of ignorance and conflict differently: A high degree of ignorance indicates that the reputation system has little information about the requested statement and suggests searching more relevant statements, if possible. A high degree of conflict, however, shows that the requested statement H is controversial. This suggests that the requester should verify whether the trust and authenticity assignments he made and that cause the conflict are correct.

Continuous trust value can be condensed to a single, linearized trust value w , if desired:

$$w' = b + W_i i + W_d d + W_c c \quad (5.4)$$

$$w = \max(0, \min(w', 1)) \quad (5.5)$$

The parameters W_i , W_d and W_c represent weights for the degrees of ignorance, disbelief and conflict, e. g., $W_i = 0.5$, $W_d = -1$ and $W_c = 0$. They can be chosen according to the preferences of the application and allow for rather optimistic or rather pessimistic behavior in the cases of uncertainty and conflict. Then the range for w is limited to $w \in [0, 1]$.

5.4.2 Deterministic Operators

Inference rules (section 5.5) define the logic of reputation systems, i. e., whether opinions can be combined and how the resulting reputation value depends on the trust values of the first-hand trust relations. In the following, we propose deterministic operators for conjunction, disjunction, negation, recommendation and consensus for the formulation of inference rules. As we favor a computation approach with probabilistic initial view it is sufficient to define these operators for discrete trust values. In section 5.6, we show how these deterministic operators can be used with continuous trust values.

To find the truth tables for the discrete trust values we proceed as follows: We represent the discrete trust values t'_x and t'_y of the input trust relations (H_x and H_y) as sets of propositions according to Table 5.3 (e. g., H_x^+ , H_y^-). For each operator we define from which combinations of the input propositions we can infer propositions for the output reputation value (H_z^+ , H_z^-). Finally, we interpret the set of output propositions as the discrete reputation value t'_z of the derived trust statement H_z .

Interestingly, this approach leads to the same truth tables for the conjunction, disjunction and negation operators as Belnap's paraconsistent four-valued logic [Bel75] although Belnap derived these operators in a different approach from a bilattice. Belnap's logic does not provide recommendation and consensus operators though.

5.4.2.1 Negation Operator

The negation operator computes the reputation value of the opposite of a trust statement. It is denoted by $t'_z = \neg t'_x$ and $\frac{\neg H_x}{H_z}$. We can conclude that there are indications supporting H_z if we have indications refuting H_x , and vice versa:

$$\frac{\neg H_x}{H_z} \Leftrightarrow \frac{H_x^+}{H_z^-}, \frac{H_x^-}{H_z^+}$$

The truth table of the negation operator is shown in Figure 5.10.

\wedge	+	\emptyset	-	\pm
+	+	\emptyset	-	\pm
\emptyset	\emptyset	\emptyset	-	-
-	-	-	-	-
\pm	\pm	-	-	\pm

\vee	+	\emptyset	-	\pm
+	+	+	+	+
\emptyset	+	\emptyset	\emptyset	+
-	+	\emptyset	-	\pm
\pm	+	+	\pm	\pm

\neg	
+	-
\emptyset	\emptyset
-	+
\pm	\pm

Figure 5.10: Our deterministic conjunction, disjunction and negation operators

5.4.2.2 Conjunction Operator

The conjunction operator for deterministic trust values corresponds to the logical *AND*-operation and is denoted by $t'_z = t'_x \wedge t'_y$. The conjunction of trust statements in inference rules is denoted accordingly by $\frac{H_x \wedge H_y}{H_z}$. We can conclude that there are indications supporting H_z if we have supporting indications for both H_x and H_y . Similarly, we can conclude that there are indications refuting H_z if we have indications refuting H_x or H_y :

$$\frac{H_x \wedge H_y}{H_z} \Leftrightarrow \frac{H_x^+}{H_z^+}, \frac{H_y^+}{H_z^+}, \frac{H_x^-}{H_z^-}, \frac{H_y^-}{H_z^-}$$

According to the procedure described in subsection 5.4.2, we can now derive the truth table of the conjunction operator (see Figure 5.10) from these two statements.

Note that the conjunction of *conflict* with *disbelief* results in *disbelief* because either H_x^- or H_y^- is sufficient to justify H_z^- . It is interesting that the conjunction of *conflict* with *ignorance* results in *disbelief*, too. *Conflict* for t'_x combined with *ignorance* for t'_y for example means that we can justify H_x^+ and H_x^- . H_x^- allows us to conclude H_z^- , but H_x^+ does not allow any conclusion without H_y^+ , so that we obtain *disbelief*. The situation is different in the case of conjunction of *conflict* with *belief* which allows the justification of both H_z^+ and H_z^- and thus results in *conflict*.

5.4.2.3 Disjunction Operator

Similarly, the disjunction operator corresponds to the logical *OR*-operation and is denoted by $t'_z = t'_x \vee t'_y$. The disjunction of trust statements is denoted accordingly by $\frac{H_x \vee H_y}{H_z}$. We can conclude that there are indications supporting H_z if we have indications supporting H_x or H_y . Similarly, we can conclude that there are refuting indications for H_z if we have refuting indications for both H_x and H_y :

$$\frac{H_x \vee H_y}{H_z} \Leftrightarrow \frac{H_x^+}{H_z^+}, \frac{H_y^+}{H_z^+}, \frac{H_x^-}{H_z^-}, \frac{H_y^-}{H_z^-}$$

The truth table of the disjunction operator is shown in Figure 5.10. The disjunction of *conflict* with *ignorance* or *belief* results in *belief* because either H_x^+ or H_y^+ is sufficient to justify H_z^+ . It is not possible to justify H_z^- because this would require both H_x^- and H_y^- . The disjunction of *conflict* with *disbelief* allows the justification of both H_z^+ and H_z^- and results thus in *conflict*.

5.4.2.4 Consensus Operator

The consensus operator is used to combine the trust values of two distinct opinions (t'_x and t'_y) that refer to the identical trust statement $H_x = H_y = H_z$. It calculates the cumulative reputation value, which is denoted by $t'_z = t'_x \oplus t'_y$. This combination of trust statements is denoted by $\frac{H_x \oplus H_y}{H_z}$, but this is usually not necessary because the consensus operator is applied implicitly whenever an inference rules allows deriving an already existing trust statement. The consensus operator is defined as follows:

$$\frac{H_x \oplus H_y}{H_z} \Leftrightarrow \frac{H_x^+}{H_z^+}, \frac{H_x^-}{H_z^-}, \frac{H_y^+}{H_z^+}, \frac{H_y^-}{H_z^-} \quad (5.6)$$

We can conclude that there are indications supporting H_z (or refuting H_z) if at least one opinion has indications supporting H_z (or refuting H_z respectively), i. e., it is sufficient to unify the two sets representing the discrete trust values. The truth table of the consensus operator is shown in Figure 5.11 (left). Combining a trust value with *ignorance* or with an identical trust value does not change the trust value. Mixing *belief* and *disbelief* results in *conflict*. Conflicting trust values remain conflicting when combined with other trust values.

\oplus	+	\emptyset	-	\pm
+	+	+	\pm	\pm
\emptyset	+	\emptyset	-	\pm
-	\pm	-	-	\pm
\pm	\pm	\pm	\pm	\pm

$t'_x \otimes t'_y$	t'_x			
	+	\emptyset	-	\pm
t'_y	+	\emptyset	\emptyset	+
	\emptyset	\emptyset	\emptyset	\emptyset
	-	-	\emptyset	-
	\pm	\pm	\emptyset	\pm

\odot	+	\emptyset	-	\pm
+	+	\emptyset	-	\pm
\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
-	-	\emptyset	\emptyset	-
\pm	\pm	\emptyset	-	\pm

Figure 5.11: Consensus, recommendation and authentication operator truth tables

5.4.2.5 Recommendation Operator

The recommendation operator (\otimes) is used to concatenate two trust statements or a trust with an authenticity statement. It is reasonable for a user to adopt the opinions of trustworthy entities. However, it is not reasonable (it is in fact even dangerous) to assume that untrustworthy (malicious or incompetent) entities always tell the opposite of the truth. Instead, opinions of untrustworthy entities should be ignored. Therefore, we do not draw any conclusions from H_x^- . The operator is thus defined as follows:

$$\frac{H_x \otimes H_y}{H_z} \Leftrightarrow \frac{H_x^+ \ H_y^+}{H_z^+}, \frac{H_x^+ \ H_y^-}{H_z^-} \quad (5.7)$$

This reads as follows: H_z follows from a combination of H_x and H_y with the recommendation operator. If there are supporting indications for H_x and for H_y , then infer H_z^+ . If there are supporting indications for H_x and refuting indications for H_y , then infer H_z^- . Figure 5.11 (middle) shows the corresponding “truth table”.

5.4.2.6 Authentication Operator

The authentication operator (\odot) is used to reason with two authenticity relations between entities, descriptions and public keys:

$$\frac{H_x \odot H_y}{H_z} \Leftrightarrow \frac{H_x^+ \ H_y^+}{H_z^+}, \frac{H_x^+ \ H_y^-}{H_z^-}, \frac{H_x^- \ H_y^+}{H_z^-} \quad (5.8)$$

The operator definition can be understood as follows: Assume H_x and H_y represent statements like “A and B belong together” and “B and C belong together”, respectively. If we have supporting indications for both statements, then this supports that A and C belong together (H_z). If we have indications that A and B belong together but that B does not belong to C, then we conclude that A does not belong to C either. If neither A belongs to B nor does B belong to C, then we can draw no conclusion about A and C. Figure 5.11 (right) shows the corresponding truth table.

5.4.2.7 Properties of the Deterministic Operators

The conjunction, disjunction and negation operators are identical to Belnap’s operators [Bel75]. Therefore the standard classical properties hold, i. e., involution ($\neg(\neg H) = H$), commutativity ($H_1 \wedge H_2 = H_2 \wedge H_1$, $H_1 \vee H_2 = H_2 \vee H_1$), associativity ($(H_1 \wedge H_2) \wedge H_3 = H_1 \wedge (H_2 \wedge H_3)$, $(H_1 \vee H_2) \vee H_3 = H_1 \vee (H_2 \vee H_3)$), distributivity ($H_1 \wedge (H_2 \vee H_3) = (H_1 \wedge H_2) \vee (H_1 \wedge H_3)$, $H_1 \vee (H_2 \wedge H_3) = (H_1 \vee H_2) \wedge (H_1 \vee H_3)$) and the De Morgan laws ($\neg(H_1 \wedge H_2) = \neg H_1 \vee \neg H_2$, $\neg(H_1 \vee H_2) = \neg H_1 \wedge \neg H_2$).

Moreover we find that consensus and authentication are commutative ($H_1 \oplus H_2 = H_2 \oplus H_1$, $H_1 \odot H_2 = H_2 \odot H_1$), consensus, recommendation and authentication are associative ($(H_1 \oplus H_2) \oplus H_3 = H_1 \oplus (H_2 \oplus H_3)$, $(H_1 \otimes H_2) \otimes H_3 = H_1 \otimes (H_2 \otimes H_3)$, $(H_1 \odot H_2) \odot H_3 = H_1 \odot (H_2 \odot H_3)$), and that all operators are distributive over

consensus $(\neg(H_1 \oplus H_2) = \neg H_1 \oplus \neg H_2, H_1 \wedge (H_2 \oplus H_3) = (H_1 \wedge H_2) \oplus (H_1 \wedge H_3), H_1 \vee (H_2 \oplus H_3) = (H_1 \vee H_2) \oplus (H_1 \vee H_3), H_1 \otimes (H_2 \oplus H_3) = (H_1 \otimes H_2) \oplus (H_1 \otimes H_3), (H_1 \oplus H_2) \otimes H_3 = (H_1 \otimes H_3) \oplus (H_2 \otimes H_3), H_1 \odot (H_2 \oplus H_3) = (H_1 \odot H_2) \oplus (H_1 \odot H_3))$.

The distributivity property ensures that the resulting reputation value does not depend on the order in which the inference rules are applied. This is very important to ensure consistency in trust graphs with loops and intersecting trust paths.

5.5 Inference Rules

The inference rules specify which conclusions the reputation system can draw from a set of given trust and authenticity propositions.

5.5.1 Transitive Trust Inference Rule

This inference rule describes the *transitivity* property of trust statements. It defines in which cases two trust statements for the same capability r can be combined with the recommendation operator in order to derive a new trust statement from the trustor of the first statement (A) to the trustee of the second statement (C). The trustor A can be an entity (E_A) or a public key (K_A). The second statement can be a trust statement or a trust certificate, i. e., B can be an entity (E_B) or a public key (K_B). The final trustee C can be an entity (E_C), a public key (K_C) or a description (D_C).

$$\frac{\text{Trust}(A, B, r, h + l_2, l_1) \otimes \text{Trust}(B, C, r, h, l_2)}{\text{Trust}(A, C, r, h, l_1 + l_2)} \quad (5.9)$$

This inference rule differs from other proposed transitive trust inference rules in that it allows the combination of trust statements only if the number of recommendation hops matches: The number of recommendation hops of the first statement must equal the sum of the recommendation hops plus the chain length of the second statement. The chain length of the resulting statement is the sum of the chain lengths of the input statements. This ensures that the recommendation hop value of the trust statements decreases by one throughout the chain of first-hand trust relations (e. g., $h = 2, h = 1, h = 0$).

The example in Figure 5.12 illustrates the inference rule. The transitive trust inference rule allows to combine $H_1^+ = \text{Trust}^+(E_A, E_B, r, 2, 1)$ with $H_2^+ = \text{Trust}^+(E_B, E_C, r, 1, 1)$ to $H_4^+ = \text{Trust}^+(E_A, E_C, r, 1, 2)$ and then H_4^+ with $H_3^- = \text{Trust}^-(E_C, E_D, r, 0, 1)$ to $H_5^- = \text{Trust}^-(E_A, E_D, r, 0, 3)$.

5.5.2 Trust in Entities, Keys and Descriptions

A number of simple rules allow to infer from trust assigned to an entity to trust assigned to the holder of a key and to trust assigned to an entity identified by a description, and vice versa. If an entity is trustworthy, then the holder of a key that

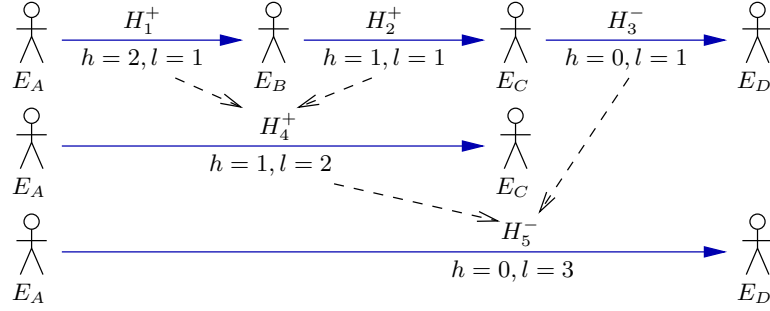


Figure 5.12: Example for application of the transitive trust inference rule

belongs to this entity is trustworthy, too, and vice versa:

$$\frac{\text{Auth}(E_A, K_C, E_C) \otimes \text{Trust}(E_A, E_C, r, h, l)}{\text{Trust}(E_A, K_C, r, h, l)} \quad (5.10)$$

$$\frac{\text{Auth}(E_A, K_C, E_C) \otimes \text{Trust}(E_A, K_C, r, h, l)}{\text{Trust}(E_A, E_C, r, h, l)} \quad (5.11)$$

If an entity is trustworthy, then the entity identified by a description that belongs to this entity is trustworthy, too, and vice versa:

$$\frac{\text{Auth}(E_A, D_C, E_C) \otimes \text{Trust}(E_A, E_C, r, h, l)}{\text{Trust}(E_A, D_C, r, h, l)} \quad (5.12)$$

$$\frac{\text{Auth}(E_A, D_C, E_C) \otimes \text{Trust}(E_A, D_C, r, h, l)}{\text{Trust}(E_A, E_C, r, h, l)} \quad (5.13)$$

If the holder of a key is trustworthy, then the entity identified by a description that belongs to this key holder is trustworthy, too, and vice versa. This applies to trust relations and trust certificates:

$$\frac{\text{Auth}(E_A, K_C, D_C) \otimes \text{Trust}(E_A, K_C, r, h, l)}{\text{Trust}(E_A, D_C, r, h, l)} \quad (5.14)$$

$$\frac{\text{Auth}(E_A, K_C, D_C) \otimes \text{Trust}(E_A, D_C, r, h, l)}{\text{Trust}(E_A, K_C, r, h, l)} \quad (5.15)$$

$$\frac{\text{Auth}(K_A, K_C, D_C) \otimes \text{Trust}(K_A, K_C, r, h, l)}{\text{Trust}(K_A, D_C, r, h, l)} \quad (5.16)$$

$$\frac{\text{Auth}(K_A, K_C, D_C) \otimes \text{Trust}(K_A, D_C, r, h, l)}{\text{Trust}(K_A, K_C, r, h, l)} \quad (5.17)$$

5.5.3 Local Authenticity Inference Rule

If an entity E_A has *partial* knowledge about whether an entity E_B is the holder of a key K_B , whether a description D_B refers to the entity E_B or whether the description D_B refers to the holder of the key K_B , then it can draw further conclusions about the trust values of the authenticity statements between E_B , K_B and D_B . If the trust values of two corresponding authenticity relations are known, then the trust value of the third authenticity relation can be derived with the authentication operator:

$$\frac{\text{Auth}(E_A, K_C, D_C) \odot \text{Auth}(E_A, K_C, E_C)}{\text{Auth}(E_A, D_C, E_C)} \quad (5.18)$$

$$\frac{\text{Auth}(E_A, K_C, D_C) \odot \text{Auth}(E_A, D_C, E_C)}{\text{Auth}(E_A, K_C, E_C)} \quad (5.19)$$

$$\frac{\text{Auth}(E_A, K_C, E_C) \odot \text{Auth}(E_A, D_C, E_C)}{\text{Auth}(E_A, K_C, D_C)} \quad (5.20)$$

5.5.4 Authenticity Inference with Authenticity Confirmation

If a trustor (E_A or K_A) trusts a trustee (E_B or K_B) to issue only correct authenticity relations or identity certificates (property r_{PKI}), then the trustor can conclude that authenticity relations or identity certificates of the trustee are correct:

$$\frac{\text{Trust}(E_A, E_B, r_{\text{PKI}}, 0, l) \otimes \text{Auth}(E_B, K_C, D_C)}{\text{Auth}(E_A, K_C, D_C)} \quad (5.21)$$

$$\frac{\text{Trust}(E_A, K_B, r_{\text{PKI}}, 0, l) \otimes \text{Auth}(K_B, K_C, D_C)}{\text{Auth}(E_A, K_C, D_C)} \quad (5.22)$$

$$\frac{\text{Trust}(K_A, K_B, r_{\text{PKI}}, 0, l) \otimes \text{Auth}(K_B, K_C, D_C)}{\text{Auth}(K_A, K_C, D_C)} \quad (5.23)$$

5.5.5 Uniqueness Conditions

Two further conclusions can be drawn from the condition that each public key has only one holder and that each description refers to only one entity. If A knows that E_B is the holder of K_B , then it can infer that all other entities are not the holder of K_B . Similarly, if A knows that E_B has the description D_B , then it can infer that all other entities do not have the description D_B (A can be an entity or a key).

$$\frac{\text{Auth}^+(A, K_B, E_B)}{\text{Auth}^-(A, K_B, E_j)}, \frac{\text{Auth}^+(A, D_B, E_B)}{\text{Auth}^-(A, D_B, E_j)} \quad \forall E_j \neq E_B \quad (5.24)$$

5.6 Trust Value Computation

The reputation system collects all issued first-hand trust and authenticity opinions H_j with associated continuous trust value t_j (with $c_j = 0$). Users can then send

requests in the form of a standard form trust statement or an authenticity statement to the reputation system. The reputation system then processes all collected opinions. It applies the inference rules to derive trust and authenticity statements and it computes the resulting continuous trust value t_0 of the requested statement H_0 from the trust values of the relevant first-hand statements. As the components of the continuous first-hand trust values (b , i and d) represent probabilities, we define the resulting trust value by a random experiment and propose different algorithms for the computation of the resulting trust value.

5.6.1 Probabilistic Model for the Trust Value Computation

The components of the computed *resulting* trust value $t_0 = (b_0, i_0, d_0, c_0)$ for H_0 are computed from the combination of all available first-hand opinions with the inference rules under the assumption that the trust values of the opinions of the requestor are correct. In short, b_0 is the computed lower bound for the probability that the combination of the available first-hand opinions leads to the conclusion that H_0 must be true (but not that H_0 must be false). Similarly, d_0 is the computed lower bound for the probability that the combination of the available first-hand opinions leads to the conclusion that H_0 must be false (but not that H_0 must be true). The degree of conflict c_0 is the computed probability that the combination of the first-hand opinions leads to the contradicting conclusion that H_0 must be both true *and* false at the same time. The degree of ignorance is the remaining probability $i_0 = 1 - b_0 - d_0 - c_0$.

The following description of a random experiment provides a more detailed definition for t_0 : We assume that the reputation system has collected J first-hand opinions, i. e., the statements H_j ($j = 1, 2, \dots, J$) with associated continuous trust values $t_j = (b_j, i_j, d_j, 0)$. For each first-hand statement H_j choose a *discrete* confidence value t'_j from $\{+, \emptyset, -\}$ according to the weights b_j , i_j and d_j , i. e., choose $t'_j = +$ with probability b_j , $t'_j = \emptyset$ with probability i_j and $t'_j = -$ with probability d_j . Statements with the discrete trust value *ignorance* don't contribute knowledge and can be discarded⁶. Each remaining first-hand statement H_j with associated discrete trust value t'_j corresponds to a set of first-hand propositions according to Table 5.3.

The inference rules always operate on trust propositions in the *internal* representation. We therefore have to replace each standard-form trust statement $\text{Trust}(A, B, r, h_{\min}..h_{\max})$ by a list of single-hop trust statements in *internal form* with chain length $l = 1$: $\text{Trust}(A, B, r, h_{\min}, l)$, $\text{Trust}(A, B, r, h_{\min} + 1, l)$, \dots , $\text{Trust}(A, B, r, h_{\max}, l)$. The internal trust statements inherit their assigned discrete trust value from the standard-form trust statement. Next, we apply all inference rules (see section 5.5) to derive all (positive and negative) deducible propositions from the set of all known first-hand propositions and all already derived propositions. To get back to trust statements in standard form we conclude

⁶this optimization does not change the resulting trust value, the resulting continuous trust value t_0 nevertheless contains the correct degree of ignorance

$H_0^+ = \text{Trust}^+(A, B, r, h_{\min}..h_{\max})$ if we have been able to derive a proposition $H_{0,h}^+ = \text{Trust}^+(A, B, r, h, l)$ with $h_{\min} \leq h \leq h_{\max}$. Similarly, we conclude H_0^- if we have been able to derive a proposition $H_{0,h}^-$.

To obtain the resulting continuous trust value of a requested trust or authenticity statement we compute the probability that the random experiment leads to a set of first-hand propositions from which we can derive positive and negative propositions for the requested statement H_0 . The components of the resulting trust value $t_0 = (b_0, i_0, d_0, c_0)$ are defined as follows: b_0 is the probability that H_0^+ (but not H_0^-) can be derived and d_0 is the probability that H_0^- (but not H_0^+) can be derived. The probability that neither H_0^+ nor H_0^- can be derived is i_0 , and c_0 is the probability that both H_0^+ and H_0^- can be derived.

In contrast to other trust models (e. g., [Jøs97, HKL00, JH05, Koh07]) we propose *not to eliminate* the degree of conflict, not only to avoid counter-intuitive effects of re-normalizations but also because it provides valuable information to the requesting user or application (see subsection 5.4.1).

5.6.2 Approximation and Exact Computation Algorithms

This section presents different possibilities to implement the computation of an approximation or of the exact value of the resulting continuous confidence value t_0 according to subsection 5.6.1. All exact algorithms return the same resulting trust value t_0 , but differ in computation time. The result of the approximation gets arbitrarily close to the exact result if the number of iterations is sufficiently large.

To keep the computation time small we recommend for all algorithms to *pre-compute all possible paths*: We first set up a “superposition” of possible first-hand propositions. For each statement H_j with continuous trust value $t_j = (b_j, i_j, d_j, 0)$ we select H_j^+ if $b_j > 0$ and we select (possibly in addition) H_j^- if $d_j > 0$. Then we translate all trust propositions into the internal form, apply all inference rules and record the dependencies, i. e., we trace which sets of first-hand propositions (premises) allow to derive which conclusions. Each set of first-hand propositions that allows to (directly or indirectly) derive the positive requested proposition H_0^+ is called a *positive path* for H_0 , each set that allows to derive the negative proposition H_0^- a *negative path* for H_0 . Next, we select the set of positive paths and the set of negative paths for H_0 and minimize these paths, i. e., we remove all paths that contain at least one other path in the set. We finally obtain the set of minimal positive paths $A^+ = \{a_1^+, a_2^+, \dots, a_{k^+}^+\}$ and the set of minimal negative paths $A^- = \{a_1^-, a_2^-, \dots, a_{k^-}^-\}$.

5.6.2.1 Approximation with Monte-Carlo Simulation

An obvious approach to determine an *approximation* for the resulting trust value is to run the described random experiment N times and to count in how many experiments the selected set of first-hand propositions contains at least one positive (but no negative) path (n_b), no paths (n_i), at least one negative (but no positive)

path (n_d) or both positive and negative paths (n_c). The approximation for the confidence value is $\bar{t}_0 = \frac{1}{N}(n_b, n_i, n_d, n_c)$. The choice of N allows to adjust the trade-off between precision and computation time.

5.6.2.2 Possible Worlds Algorithm

An simple algorithm to compute the exact value is to go through the list of all possible combinations of first-hand propositions (so-called *possible worlds*), to compute the probability of each of those possible worlds and to check for each world whether the set of first-hand propositions of this world contains the minimal paths. The sum of all probabilities of all worlds that contain at least one positive and at least one negative path is c_0 , b_0 is the sum of probabilities of all worlds that contain at least one positive, but no negative path, and d_0 the sum of probabilities of all worlds that contain at least one negative, but no positive path. The degree of ignorance is $i_0 = 1 - b_0 - d_0 - c_0$.

5.6.2.3 Trust Example 1: Simple Trust Chain with Possible Worlds Algorithm

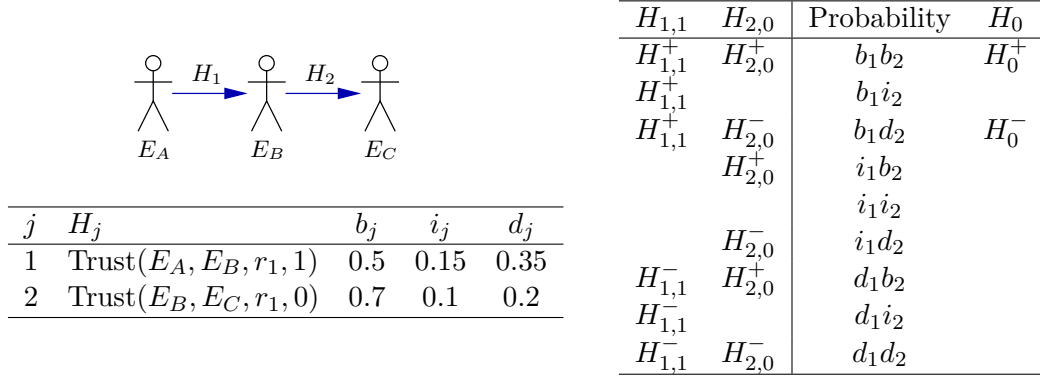


Figure 5.13: Scenario and possible worlds table of the Trust Example 1

The example scenario in Figure 5.13 (left) consists of two trust statements: H_1 is a recommendation trust statement for one recommendation hop ($h = 1$), and H_2 is a functional trust statement ($h = 0$). E_A wants to compute the resulting functional trustworthiness of E_C ($H_0 = \text{Trust}(E_A, E_C, r_1, 0)$).

First, the trust statements in standard form have to be replaced by corresponding trust statements in internal form: H_1 by $H_{1,1} = \text{Trust}(E_A, E_B, r_1, 1, 1)$ and H_2 by $H_{2,0} = \text{Trust}(E_B, E_C, r_1, 0, 1)$. Both refer to the same property r_1 , it is therefore possible to combine $H_{1,1}$ and $H_{2,0}$ with the transitive trust inference rule Equation 5.9 to the new functional trust statement $H_{0,0} = \text{Trust}(E_A, E_C, r_1, 0, 2)$: $H_{1,1}^+, H_{2,0}^+ \vdash H_{0,0}^+$ ($H_{1,1}^+$ and $H_{2,0}^+$ allow to drive $H_{0,0}^+$) and $H_{1,1}^+, H_{2,0}^- \vdash H_{0,0}^-$. Thus, there is only one positive path $a_1^+ = \{H_{1,1}^+, H_{2,0}^+\}$ and one negative path $a_1^- = \{H_{1,1}^+, H_{2,0}^-\}$ for $H_{0,0}$ and thus for H_0 .

Figure 5.13 (right) shows all possible combinations of the first-hand propositions, the probability that this world occurs and the propositions that can be derived in this world. There are no worlds in which both H_0^+ and H_0^- can be derived, thus $c_0 = 0$. H_0^+ can be derived only in the first world, therefore $b_0 = b_1b_2$. Similarly, H_0^- can be derived only in the third world, therefore $d_0 = b_1d_2$. The degree of ignorance is the remaining probability mass $i_0 = 1 - b_0 - d_0 - c_0$. With the values in Figure 5.13 we obtain $t_0 = (0.35, 0.55, 0.1, 0)$.

5.6.2.4 Grouped Possible Worlds Algorithm

The possible worlds algorithm can be improved by subdividing the set of relevant first-hand statements into as few groups g_1, \dots, g_u as possible. Two statements, H_j and H_m , belong to the same group if and only if the following condition holds for each positive, negative and conflicting path: If the path contains a proposition for H_j (H_j^+ or H_j^-), then it must also contain a proposition for H_m (H_m^+ or H_m^-).

In the preparation step we construct for each group a list of all relevant combinations of propositions of the statements in the group. This list contains all combinations that contain exactly one proposition (i. e., either H_j^+ or H_j^-)⁷ for each statement and that is identical to the corresponding section of at least one (positive or negative) path. An additional element of this list consists of an empty set. It represents all remaining possible combinations of propositions, i. e., all combinations that contain neither H_j^+ nor H_j^- for at least one statement H_j of the group. We can subsume these combinations because they have the same effect on the derivability of propositions of H_0 . For each element of the list we compute the probability that this combination will occur (within the group) from the continuous trust values of the statements. The probability associated with the empty set is the sum of the probabilities of the contained combinations of propositions (i. e., the remaining probability). Thus, the sum of all probabilities is one.

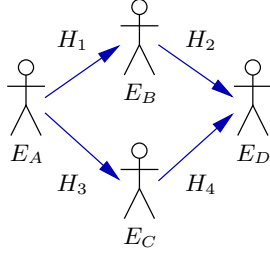
Next, in the main step, we go through all possible worlds. Each world consists of one possible combination of these prepared proposition-combinations of the groups, i. e., for each group we select one proposition-combination from the prepared list of the group. We multiply the precomputed probabilities of the chosen proposition-combinations to obtain the resulting probability of the world. Finally, we compute b_0 , i_0 , d_0 and c_0 just as in the possible worlds algorithm.

5.6.2.5 Trust Example 2: Parallel Trust Chain with Grouped Possible Worlds Algorithm

The scenario in Figure 5.14 consists of two parallel trust chains from E_A to E_D . E_A requests the trust value for the resulting functional trustworthiness of E_D ($H_0 = \text{Trust}(E_A, E_D, r_1, 0)$). The trust statements in standard form are replaced by statements in internal form: H_1 by $H_{1,1} = \text{Trust}(E_A, E_B, r_1, 1, 1)$, H_2 by $H_{2,0} = \text{Trust}(E_B, E_D, r_1, 0, 1)$, H_3 by $H_{3,1} = \text{Trust}(E_A, E_C, r_1, 1, 1)$ and H_4

⁷no combination can contain both H_j^+ and H_j^- because $c_j = 0$

5 Trust Model



j	H_j	b_j	i_j	d_j
1	$\text{Trust}(E_A, E_B, r_1, 1)$	0.8	0.15	0.05
2	$\text{Trust}(E_B, E_D, r_1, 0)$	0.7	0.1	0.2
3	$\text{Trust}(E_A, E_C, r_1, 1)$	0.9	0.1	0
4	$\text{Trust}(E_C, E_D, r_1, 0)$	0.8	0.1	0.1

Figure 5.14: Scenario of the Trust Example 2

by $H_{4,0} = \text{Trust}(E_C, E_D, r_1, 0, 1)$. We can combine $H_{1,1}$ with $H_{2,0}$ and $H_{3,1}$ with $H_{4,0}$ with the transitive trust inference rule Equation 5.9. We obtain two positive paths $A^+ = \{\{H_{1,1}^+, H_{2,0}^+\}, \{H_{3,1}^+, H_{4,0}^+\}\}$ and two negative paths $A^- = \{\{H_{1,1}^+, H_{2,0}^-\}, \{H_{3,1}^+, H_{4,0}^-\}\}$. Propositions for $H_{1,1}$ and $H_{2,0}$ appear always together in paths, the same holds for $H_{3,1}$ and $H_{4,0}$. Therefore we can divide the statements into two groups $g_1 = \{H_{1,1}, H_{2,0}\}$ and $g_2 = \{H_{3,1}, H_{4,0}\}$.

In the preparation step we set up a list for each group that contains all relevant possible combinations of the propositions and their probabilities (see Table 5.4). For each group we find three relevant combinations: one combination supports a positive path and one a negative path. The third entry with the empty set represents the remaining combinations.

Propositions g_1	Probability	Propositions g_2	Probability
$\{H_{1,1}^+, H_{2,0}^+\}$	$b_1 b_2$	$\{H_{3,1}^+, H_{4,0}^+\}$	$b_3 b_4$
$\{H_{1,1}^+, H_{2,0}^-\}$	$b_1 d_2$	$\{H_{3,1}^+, H_{4,0}^-\}$	$b_3 d_4$
$\{\}$	$1 - b_1 b_2 - b_1 d_2$	$\{\}$	$1 - b_3 b_4 - b_3 d_4$

Table 5.4: Preparation step for the groups in the Trust Example 2

g_1	g_2	Probability	H_0
$\{H_{1,1}^+, H_{2,0}^+\}$	$\{H_{3,1}^+, H_{4,0}^+\}$	$b_1 b_2 b_3 b_4$	H_0^+
$\{H_{1,1}^+, H_{2,0}^+\}$	$\{H_{3,1}^+, H_{4,0}^-\}$	$b_1 b_2 b_3 d_4$	H_0^+, H_0^-
$\{H_{1,1}^+, H_{2,0}^+\}$	$\{\}$	$b_1 b_2 (1 - b_3 b_4 - b_3 d_4)$	H_0^+
$\{H_{1,1}^+, H_{2,0}^-\}$	$\{H_{3,1}^+, H_{4,0}^+\}$	$b_1 d_2 b_3 b_4$	H_0^+, H_0^-
$\{H_{1,1}^+, H_{2,0}^-\}$	$\{H_{3,1}^+, H_{4,0}^-\}$	$b_1 d_2 b_3 d_4$	H_0^-
$\{H_{1,1}^+, H_{2,0}^-\}$	$\{\}$	$b_1 d_2 (1 - b_3 b_4 - b_3 d_4)$	H_0^-
$\{\}$	$\{H_{3,1}^+, H_{4,0}^+\}$	$(1 - b_1 b_2 - b_1 d_2) b_3 b_4$	H_0^+
$\{\}$	$\{H_{3,1}^+, H_{4,0}^-\}$	$(1 - b_1 b_2 - b_1 d_2) b_3 d_4$	H_0^-
$\{\}$	$\{\}$	$(1 - b_1 b_2 - b_1 d_2)(1 - b_3 b_4 - b_3 d_4)$	

Table 5.5: Trust value computation in the Trust Example 2

To compute the resulting trust value t_0 for H_0 we set up Table 5.5 with all $3 \cdot 3 = 9$ possible combinations of the entries in the lists (possible worlds), the probabilities of each world and the derivable propositions for H_0 . Then we add the probabilities and obtain $b_0 = b_1 b_2 b_3 b_4 + b_1 b_2 (1 - b_3 b_4 - b_3 d_4) + (1 - b_1 b_2 - b_1 d_2) b_3 b_4$, $i_0 = (1 - b_1 b_2 - b_1 d_2)(1 - b_3 b_4 - b_3 d_4)$, $d_0 = b_1 d_2 b_3 d_4 + b_1 d_2 (1 - b_3 b_4 - b_3 d_4) + (1 - b_1 b_2 - b_1 d_2) b_3 d_4$ and $c_0 = b_1 b_2 b_3 d_4 + b_1 d_2 b_3 b_4$. With the values in Figure 5.14 we obtain $t_0 = (0.7112, 0.0532, 0.07, 0.1656)$.

5.6.2.6 Computation with Inclusion-exclusion Formula

This algorithm computes the exact resulting trust value directly from the minimal positive and negative paths for H_0 . In addition, we need the set of minimal *conflicting paths*. Therefore we set up all possible combinations consisting of one positive and one negative path ($a_x^\pm = a_y^+ \cup a_z^-$ with $y = 1, \dots, k^+$, $z = 1, \dots, k^-$), minimize the set and obtain $A^\pm = \{a_1^\pm, a_2^\pm, \dots, a_{k^\pm}^\pm\}$ (with $k^\pm \leq k^+ k^-$). A useful optimization is to eliminate all paths that contain both H_j^+ and H_j^- (because $c_j = 0$).

First, we compute the degree of conflict c_0 from the trust values of the first-hand statements in the set of minimal paths with the inclusion-exclusion-formula ($I(A)$): c_0 is the probability that a possible world chosen according to subsection 5.6.1 will contain at least one conflicting path. Thus, we add the probabilities of all minimal paths, subtract the probabilities of all unions of two minimal paths, add the probabilities of all unions of three minimal paths, etc.:

$$\begin{aligned} c_0 = I(A^\pm) &= \sum_{n=1}^{k^\pm} (-1)^{n+1} \sum_{1 \leq j_1 < \dots < j_n \leq k^\pm} P(a_{j_1}^\pm \cup \dots \cup a_{j_n}^\pm) \\ &= \sum_{j_1=1}^{k^\pm} P(a_{j_1}^\pm) - \sum_{1 \leq j_1 < j_2 \leq k^\pm} P(a_{j_1}^\pm \cup a_{j_2}^\pm) + \dots + (-1)^{k^\pm+1} P(a_1^\pm \cup \dots \cup a_{k^\pm}^\pm) \end{aligned} \quad (5.25)$$

$P(a)$ denotes the probability that path a is contained in the set of first-hand propositions of a chosen possible world:

$$P(a) = \prod_{j: H_j^+ \in a \text{ or } H_j^- \in a} p_j \quad \text{with } p_j = \begin{cases} 0 & \text{if } H_j^+ \in a, H_j^- \in a \\ b_j & \text{if } H_j^+ \in a, H_j^- \notin a \\ d_j & \text{if } H_j^+ \notin a, H_j^- \in a \end{cases} \quad (5.26)$$

We obtain $b_0 + c_0$ with the inclusion-exclusion formula applied to the minimal positive paths, thus $b_0 = I(A^+) - c_0$. Similarly, the degree of disbelief is $d_0 = I(A^-) - c_0$ and finally we obtain $i_0 = 1 - b_0 - d_0 - c_0$.

5.6.2.7 Trust Example 3: Authenticity Verification with Inclusion-Exclusion Formula

Figure 5.15 shows an example scenario consisting of the first-hand statements H_1, \dots, H_6 with associated confidence values. Entity E_A wants to know whether entity

5 Trust Model

E_D is the holder of the key K_D and therefore requests the resulting trust value for $H_0 = \text{Auth}(E_A, K_D, E_D)$.

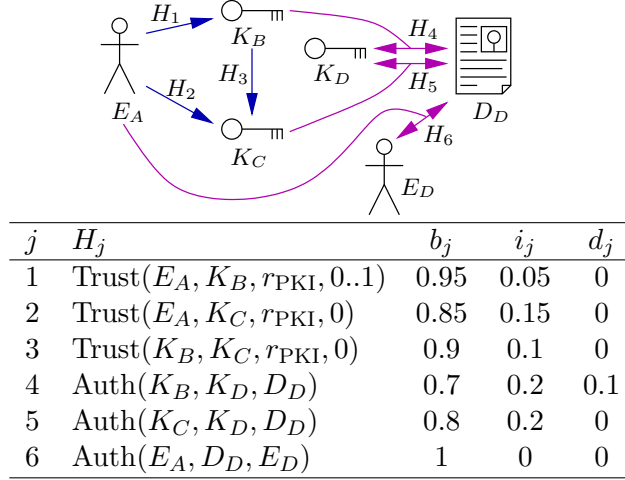


Figure 5.15: Scenario of the Trust Example 3

Proposition	Inference rule	Origin
$H_{1,0}^+ = \text{Trust}^+(E_A, K_B, r_{\text{PKI}}, 0, 1)$	-	from H_1
$H_{1,1}^+ = \text{Trust}^+(E_A, K_B, r_{\text{PKI}}, 1, 1)$	-	from H_1
$H_{2,0}^+ = \text{Trust}^+(E_A, K_C, r_{\text{PKI}}, 0, 1)$	-	from H_2
$H_{3,0}^+ = \text{Trust}^+(K_B, K_C, r_{\text{PKI}}, 0, 1)$	-	from H_3
$H_4^+ = \text{Auth}^+(K_B, K_D, D_D)$	-	from H_4
$H_4^- = \text{Auth}^-(K_B, K_D, D_D)$	-	from H_4
$H_5^+ = \text{Auth}^+(K_C, K_D, D_D)$	-	from H_5
$H_6^+ = \text{Auth}^+(E_A, D_D, E_D)$	-	from H_6
$H_7^+ = \text{Trust}^+(E_A, K_C, r_{\text{PKI}}, 0, 2)$	Equation 5.9	$H_{1,1}^+, H_{3,0}^+ \vdash H_7^+$
$H_8^+ = \text{Auth}^+(E_A, K_D, D_D)$	Equation 5.22	$H_{1,0}^+, H_4^+ \vdash H_8^+; H_{2,0}^+, H_5^+ \vdash H_8^+; H_7^+, H_5^+ \vdash H_8^+$
$H_8^- = \text{Auth}^-(E_A, K_D, D_D)$	Equation 5.22	$H_{1,0}^+, H_4^- \vdash H_8^-$
$H_0^+ = \text{Auth}^+(E_A, K_D, E_D)$	Equation 5.19	$H_6^+, H_8^+ \vdash H_0^+$
$H_0^- = \text{Auth}^-(E_A, K_D, E_D)$	Equation 5.19	$H_6^+, H_8^- \vdash H_0^-$

Table 5.6: Propositions and applied inference rules in the Trust Example 3

First, we transform the trust statements from standard form into the internal form: H_1 is transformed into $H_{1,0}^+ = \text{Trust}(E_A, K_B, r_{\text{PKI}}, 0, 1)$ and $H_{1,1}^+ = \text{Trust}(E_A, K_B, r_{\text{PKI}}, 1, 1)$, H_2 into $H_{2,0}^+ = \text{Trust}(E_A, K_C, r_{\text{PKI}}, 0, 1)$, etc. Then we create the set of propositions that represents a superposition of all possible worlds according to the trust values of the statements (see Table 5.6, $H_{1,0}^+, \dots, H_6^+$). Next, we apply the inference rules to the proposition of this set (including the

already derived propositions). The remaining rows of Table 5.6 list the derived propositions as well as the used inference rules and the premises. The transitive trust inference rule Equation 5.9 allows for example to derive the new proposition $H_7^+ = \text{Trust}^+(E_A, K_C, r_{\text{PKI}}, 0, 2)$ from $H_{1,1}^+$ and $H_{3,0}^+$. Then the minimal positive and negative paths can be determined. We find the three positive paths $\{H_1^+, H_4^+, H_6^+\}$, $\{H_2^+, H_5^+, H_6^+\}$ and $\{H_1^+, H_3^+, H_5^+, H_6^+\}$ and one negative path $\{H_1^+, H_4^-, H_6^+\}$. We can thus construct the set of minimal conflicting paths: $\{H_1^+, H_2^+, H_4^-, H_5^+, H_6^+\}$, $\{H_1^+, H_3^+, H_4^-, H_5^+, H_6^+\}$ and $\{H_1^+, H_4^+, H_4^-, H_6^+\}$. The last path can be eliminated since $c_4 = 0$.

Next we compute the degrees of conflict, belief and disbelief with the inclusion-exclusion formula: $c_0 = b_1 b_2 d_4 b_5 b_6 + b_1 b_3 d_4 b_5 b_6 - b_1 b_2 b_3 d_4 b_5 b_6 = 0.07486$, $b_0 = b_1 b_4 b_6 + b_2 b_5 b_6 + b_1 b_3 b_5 b_6 - b_1 b_2 b_4 b_5 b_6 - b_1 b_3 b_4 b_5 b_6 - b_1 b_2 b_3 b_5 b_6 + b_1 b_2 b_3 b_4 b_5 b_6 - c_0 = 0.92358 - c_0 = 0.84872$ and $d_0 = b_1 d_4 b_6 - c_0 = 0.095 - c_0 = 0.02014$. The degree of ignorance is $i_0 = 1 - b_0 - d_0 - c_0 = 0.05628$. Thus, the resulting trust value for H_0 is $t_0 = (0.84872, 0.05628, 0.02014, 0.07486)$.

5.7 Evaluation of the Proposed Trust Model

5.7.1 Computation Time of the Proposed Computation Algorithms

Computation time is a very (although not the most) important issue for reputation systems. The computation time to find the minimal paths appears to be uncritical because it is possible to check the inference rules efficiently and because the paths can be computed incrementally and in advance. Furthermore, the paths usually remain unchanged when the trust values of existing opinions are updated.

The number of possible worlds to consider in the possible worlds algorithm increases exponentially with the number of *relevant first-hand statements*. It is therefore applicable if the number of relevant statements is small. It is important to emphasize that the computation time depends only on the number of *relevant* statements or paths, not on the *total* number. It is sufficient to consider only statements that are issued by the requester or by authentic entities or keys that have been found to be trustworthy. Moreover, we can ignore all statements that are not part of a valid path, i. e., that do not contribute to answer the trust or authenticity request. Furthermore, most trust chains will not be longer than two or three statements. Therefore, the number of relevant statements or paths will usually be reasonably small. Although a trust and authenticity network similar to the PGP/GnuPG web of trust [ACGW99] can contain more than 100 000 trust and authenticity statements, the number of statements that are directly or indirectly (via valid paths) related to the requester will probably be below 100, and the number of statements that are part of valid paths from the requester to the requested statement is likely to be not higher than 10 or 20 in typical scenarios.

The number of possible worlds in the grouped possible worlds algorithm increases exponentially with the number of *groups*. Thus, the computation time can be reduced significantly if the statements can be grouped. Even large scenarios can be

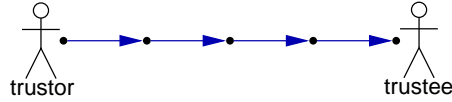


Figure 5.16: Scenario of the simple chain example

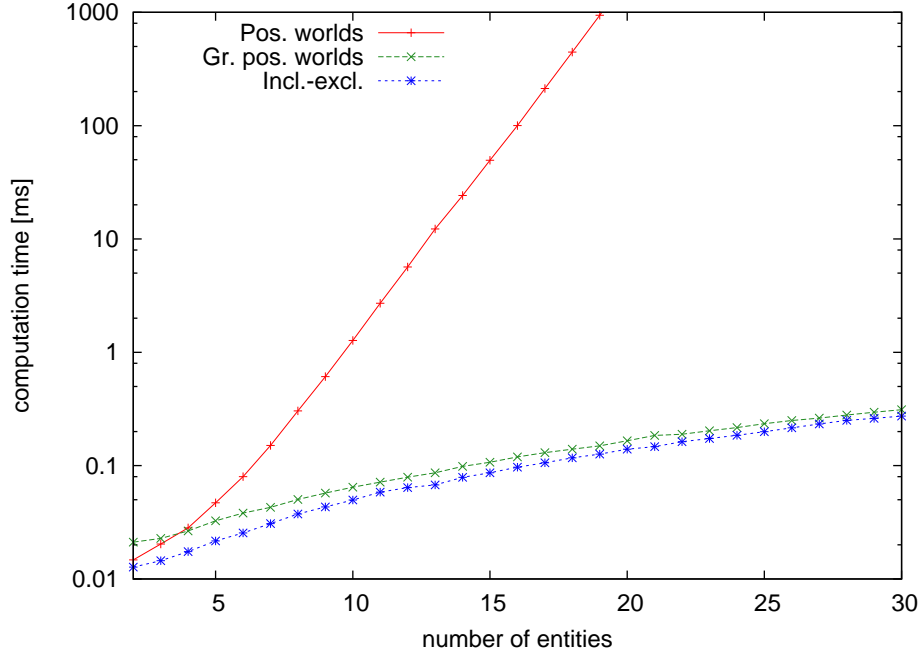


Figure 5.17: Computation time in the simple chain example

evaluated efficiently as long as the relevant statements can be subdivided into a small number of groups. In the inclusion-exclusion algorithm the number of summands increases exponentially with the number of relevant *paths*. This algorithm is therefore well suited for all scenarios with a small number of paths, even if the number of statements is large.

We illustrate the influence of the scenario (i. e., the number of relevant statements, paths and groups) on the computation time of the algorithms on two examples⁸. The scenarios are constructed in order to emphasize the large influence on the computation time and are not meant to be representative examples. For simplicity the scenarios consist only of trust statements between entities and refer to the same capability r . All trust values contain degrees of belief, ignorance and disbelief ($b > 0, i > 0, d > 0$).

The scenario with e entities in Figure 5.16 consists of a simple chain and $e - 1$ trust statements with $h = 0..e$ recommendation hops. The possible worlds algorithm has

⁸implementation in Java 1.6; measurements on Intel Pentium M with 1.73 GHz

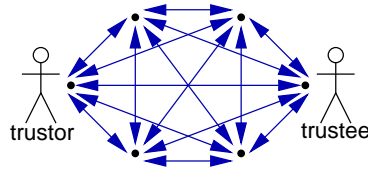


Figure 5.18: Scenario of the full mesh example

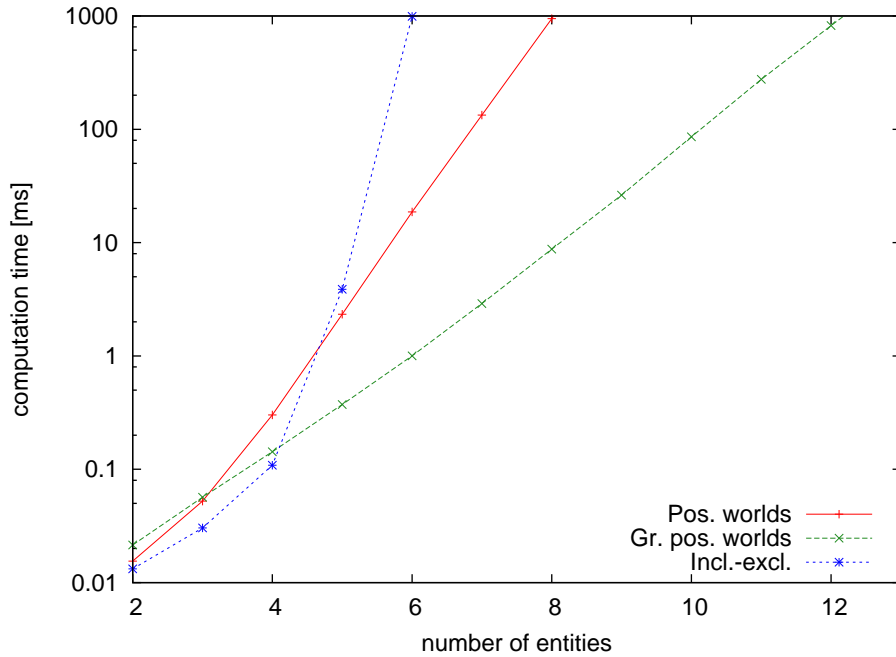


Figure 5.19: Computation time in the full mesh example

to evaluate 3^{e-1} worlds. The scenario contains one positive and one negative path, therefore the inclusion-exclusion algorithm has to compute only two summands. The grouped possible world algorithm creates one group with three possible proposition-combinations: the positive path leads to belief, the negative path to disbelief, all other combinations lead to ignorance. It thus has to evaluate only three worlds. The diagram in Figure 5.17 shows that the computation time of the possible world algorithm increases exponentially with the number of trust statements, whereas the computation time of the other algorithms increases linearly and thus remains insignificant even for long trust chains.

The scenario in Figure 5.18 is a full mesh. All entities trust each other for $h = 0.1$ hops. The number of relevant statements is $2e - 3$, the number of positive and negative paths is $e - 1$ and the number of conflicting paths is $(e - 1)(e - 2)$. Thus, the computation time of the possible worlds algorithm increases slower than of the inclusion-

exclusion algorithm because the number of conflicting paths increases faster than the number of relevant statements (Figure 5.19). The grouped possible worlds algorithm subdivides the statements into $e - 1$ groups, which reduces the number of possible worlds from 3^{2e-3} to 3^{e-1} worlds. Therefore the computation time remains acceptable for a larger number of entities than with the other algorithms.

The computation time heavily depends on the scenario. It is therefore difficult to give a general recommendation for one of the algorithms. It is possible that one algorithm outperforms an other by orders of magnitude in one scenario, but is much slower in an other scenario. The presented results and measurements in other scenarios suggest that the grouped possible worlds algorithm is in most scenarios the fastest (or at least close to the fastest) algorithm. However, further investigations are necessary.

An estimation for the computation time of the algorithms can be computed from the number of relevant statements, paths and groups. If the expected computation of all exact algorithms exceeds an acceptable limit, then a Monte-Carlo simulation can be used to compute an approximation. The number of iterations can be chosen according to the required accuracy and the acceptable computation time.

5.7.2 Comparison with Other Trust Models

The model of Maurer [Mau96] does not allow to express degrees of disbelief. Therefore, conflict can never occur. In all scenarios in which Maurer's model can be applied it produces the same resulting trust values as our model. Subjective Logic [Jøs97] can only be applied if the network is a directed series-parallel graph (e.g., Trust Examples 1 and 2, but not Trust Example 3). Credential Networks [JH05] can be applied only if at least one component of the trust value (b_j , i_j or d_j) of each first-hand trust value is zero. Subjective Logic and Credential Networks produce the same results as our model in all cases in which the models and their computation approaches can be applied and in which *no conflict* occurs (e.g., in the Trust Example 1). If conflicts are possible (e.g., in the Trust Examples 2 and 3), then the results generally differ from the results of our model because these models eliminate the probability mass associated with conflict.

Our model can thus be seen as an extension of Maurer's model, Subjective Logic and Credential Networks that overcomes the mentioned restrictions ($b > 0$, $i > 0$ and $d > 0$ is possible, no restriction to directed series-parallel graphs). However, we do not eliminate the degree of conflict, because this can cause counter-intuitive effects: Consider Trust Example 2 (subsubsection 5.6.2.5). If we choose $t_1 = t_2 = t_3 = t_4 = (1, 0, 0, 0)$ (full trust), then the resulting trust value is $t_0 = (1, 0, 0, 0)$, too (in all models). If t_4 changes to $t_4 = (0.01, 0, 0.99, 0)$ (almost complete distrust), then the resulting trust value in our model changes to $t_0 = (0.01, 0, 0, 0.99)$, which shows E_A that the trustworthiness of E_D is now highly disputed. However, in Subjective Logic and Credential Networks the resulting trust value does not change. This gives E_A the wrong impression that there are no trustworthy entities who distrust E_D .

5.7.3 Trade-offs and Limitations

It is important to be aware of the fact that all models for trust and authenticity relations make more or less intuitive assumptions and more or less extensive simplifications in order to keep the computational complexity on an acceptable level. The decision for an appropriate reputation system certainly depends on the intended application and is often a trade-off between precision and accuracy on the one hand and performance and practicability on the other hand. Simple trust models may be fast and easy to implement, but oversimplified models cannot capture and reflect all relevant properties of trust relations appropriately and they therefore suffer from undesirable and counterintuitive effects. With more advanced models that describe the properties of and relations between trust and authenticity statements more detailed and that process the opinions in a more sophisticated manner, it is possible to largely avoid counterintuitive effects, but this usually results in a significantly higher computational complexity.

There are numerous simple trust models, but there seems to be a lack of more advanced trust and authentication models (see subsection 5.7.2). Therefore our primary concern was to develop a precise model that provides intuitive results. Our model is nevertheless practical because the computation time can be kept low by using the approximative Monte-Carlo method (subsubsection 5.6.2.1) for complex trust and authentication networks instead of the exact algorithms.

One should also be aware of the fact that reputation systems can only operate on the basis of the information that is accessible to the system, i. e., the explicitly formulated trust and authenticity opinions. These will usually be incomplete and often outdated because users have only limited time and motivation to keep their opinion on the trustworthiness of others up to date. The computed reputation values are thus always subject to uncertainty and should therefore be used with care.

There exist several inherent limitations for reputations systems for which it is difficult or impossible to find purely technical solutions: The issued trust statements usually have to be available to other users of the system. This leads to a *privacy dilemma*, because they hereby disclose very sensitive information about their personal relationships (by disclosing whom they trust and distrust) and likings. This disclosure may even lead to the effect that users give not honest but rather *socially desirable* ratings or act for other reasons (e. g., revenge) in a *strategic* manner. Moreover, the disclosure of trust statements could affect the interpersonal relationships of the involved users. Assigning a low trust values to someone could be interpreted as a first sign of distrust and actually damage or destroy sensitive trust relations.

5.8 Reliability of Context Information

Trust always refers to entities (i. e., users, provider, ...), not to context information or data, because data cannot “behave as expected”.

It is nevertheless useful to define a quality metric for context information that describe to which degree a user can *rely* on the context information. We use the

term *reliability* to refer to the property of context information to be *reliable* in the sense that the context information is *correct*, i. e., that the referenced objects exist in the real world, that it describes the referenced objects truthfully within the limits of the indicated accuracy of the information.

All users of the system may issue statements with their (subjective) opinion about the reliability of any piece of context information in the system. A reliability statement may refer to

- the value of an attribute of an object,
- to an object as a whole or
- to an context information provider.

A reliability statement may refer to an object as a whole for example if the user believes that the object does not exist at all in the real world or that it is composed of attributes that do not belong to the same object. A reliability statement may refer to an context information provider if the user believes that all objects issued by this provider are unreliable. Users can digitally sign their issued reliability statements in order to protect the authenticity of their ratings. If a user has issued several reliability statements (e. g., one reliability statement for the attribute and another for the provider), then it is reasonable to choose the *most specific* reliability value (i. e., the reliability value for the attribute, or, if not available, the value for the object).

Reliability is expressed by a real number $v \in [0, 1]$, where $v = 0$ represents completely unreliable information and $v = 1$ completely reliable information. Reliability values $0 < v < 1$ represent a subjective estimation of the issuing user for the probability that the referenced context information is correct in the above sense.

Different users may have different opinions about the reliability of a given context information o_k^i of the data provider i . The opinion of the user j about the reliability of the attribute P of a context information o_k^i of the data provider i is expressed by $v_j(o_k^i.P)$. A component of a context-aware system can compute a *resulting reliability value* $v(o_k^i.P)$ for a given attribute of a context information $o_k^i.P$ using different strategies. It can for example

- choose the reliability value $v(o_k^i.P) = v_j(o_k^i.P)$ of the user j with the highest linearized trust value w_j , or
- compute the average reliability value weighted with the linearized trust values of the issuers $v(o_k^i.P) = \frac{\sum_j w_j v_j(o_k^i.P)}{\sum_j w_j}$.

The resulting reliability value can be used to separate reliable from unreliable information, e. g., to decide whether to use or discard a piece of information when processing a query or to choose the most likely information from a set of conflicting alternatives.

5.9 Summary

We presented a detailed model to represent trust and authenticity statements as well as trust values and we proposed an integrated approach to reason with these statements and to compute resulting trust values. The model distinguishes clearly between entities, descriptions and keys, allows multiple keys and descriptions per entity, distinguishes between functional and recommendation trust and allows to specify ranges of recommendation hops in trust statements. Trust values allow to express degrees of belief, ignorance and disbelief. The system is able to reason with conflicting opinions because the presented inference rules are based on a paraconsistent logic. The computation of the resulting trust values is based on a probability theoretical model in order to produce consistent results. In conflict-free scenarios our model is consistent with the Model of Maurer, Subjective Logic and Credential Networks, but overcomes several restrictions of these models. In conflicting scenarios we do not eliminate the degree of conflict in order to avoid counter-intuitive effects caused by re-normalizations. Furthermore we proposed different algorithms to implement the trust value computation. Although the computation time increases exponentially with the number of relevant statements, groups or paths it can be expected that an acceptable computation time can be reached in the majority of realistic scenarios. In the other cases, we propose to compute an approximation with Monte-Carlo simulations. The computed resulting trust values can be used to merge the ratings of different users about the reliability of context information and context information providers.

6 Reference Model

Until now, we have presented Degradation, Consistency, and Trust as separate issues with regard to the Quality of Context information (QoC). However, there are a number of interdependencies between them, as we will discuss in the following Section. Based on this discussion, we are presenting an integrated, quality-aware processing model for queries on context information. The processing model incorporates the mechanisms for all three aspects of Quality of Context, which we presented in the previous chapters.

6.1 Interdependencies between Uncertainty, Consistency, and Trust

An integrated, quality-aware query processing for context data requires an understanding of the relationship between the different aspects Degradation, Consistency, and Trust. For example, take the position information for an object O_8 , e.g., a bus, which is available from two providers, S_1 and S_2 . From this example, Figure 6.1 shows the following interdependencies between Degradation, Consistency, and Trust:

- *Uncertainty*: The two independent providers, S_1 and S_2 have (uncertain) position information for the object O_8 , the bus, in their model. Due to different sensors, different update protocols, etc., the two providers have slightly different values for the position – (3.4, 4.2) as opposed to (3.3, 4.1) – as well as different uncertainties attached to the values – ± 5 m and ± 4 m. Individually, each piece of (uncertain or degraded) information from the two providers can be correct in regard to the real world, i.e., the actual position of the bus.
- *Consistency*: When a piece of information (with knowledge about the uncertainty) is available from several different providers, such as S_1 and S_2 in the example, the information from the different providers can be inconsistent. For example, if each provider states that an object is within a certain area, but the two areas do not overlap, one or both must be wrong, i.e., the information of the two is inconsistent. Whether information about the same phenomenon from several sources is consistent depends on whether the different uncertain information can be correct at the same time as well as the probability for both being correct.
- *Trust/Reliability*: Individual providers can have incorrect and therefore unreliable information. There are a number of reasons for this: inexpensive and thus inaccurate sensors provide incorrect data, update protocols prioritize energy saving over the accuracy of the data, or even a malicious provider. Entities

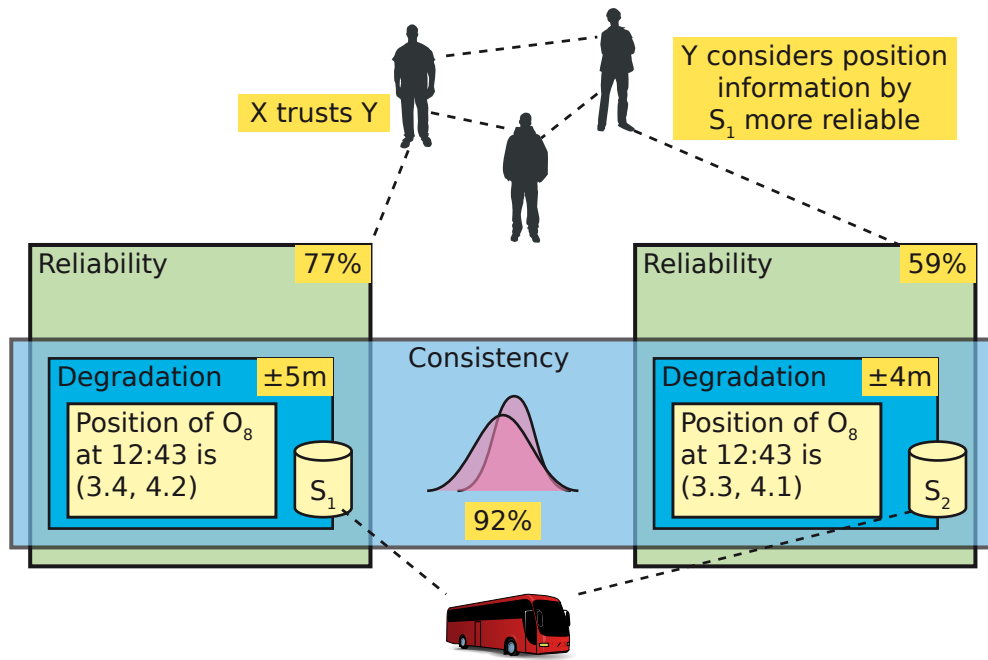


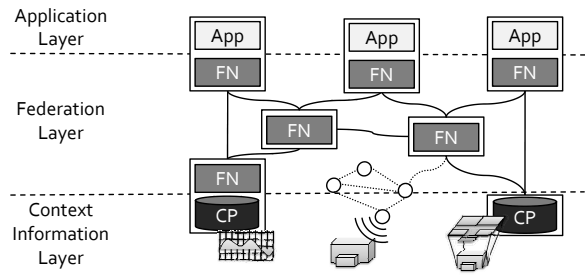
Figure 6.1: Interdependencies between Degradation, Consistency, and Trust

can thus make statements about the reliability of individual providers or pieces of information. For example, if one provider regularly offers incorrect information, the users would sooner or later doubt the reliability of this particular provider's information. This is independent from the uncertainty of the information as well as from the consistency between several providers. However, not every user can necessarily judge the reliability of a provider. Therefore, a user can trust other users' judgements about the reliability, creating a web of trust, which is used to propagate the reliability information of individual providers.

Based on these interdependencies, the following Section proposes a processing model, which incorporates these aspects of Quality of Context (QoC) into the Nexus system.

6.2 Integrated Quality-Aware Processing Model

Applications that process sensed data or integrate data from different independent data providers need to handle data with varying quality. To these applications it is crucial to measure data quality. Moreover, a measurement of quality can be beneficial for both applications and data providers. Applications can use it to exclude data that does not satisfy user needs and data providers could incorporate the quality of the provided data into their pricing policies. Often the quality of data hints at

Figure 6.2: Architecture of the Nexus system [LCG⁺09a]

the costs of providing the data. It might be more expensive in terms of energy to provide an accurate, up-to-date data value of a sensor than an imprecise, possibly outdated value. Especially context-aware applications running on resource-limited mobile devices often have to trade quality against resource-consumption. These context-aware applications are the focus of the Nexus project.

A lot of research has been done on the subject of data quality. In most cases a metric of a certain quality aspect like uncertainty is used to define quality. In the context of the Nexus project, we have investigated three different aspects of quality: uncertainty, inconsistency and reliability. In this section we integrate all three aspects of quality into a single processing model.

This section is organized as follows. Section 6.2.1 introduces the Nexus middleware and motivates the choice of the three quality aspects. We introduce operators used for formulating queries and an example scenario in Section 6.2.3. In Section 6.2.4, we explain the reasons for integrating the three quality aspects and present and evaluate a suitable processing model.

6.2.1 The Nexus System

In the Nexus project [LCG⁺09a], we provide a framework for managing global context models in an open platform, where a multitude of context data providers can integrate and share their context models. Due to the global characteristics and the high number of different context providers, our system is based on a distributed and scalable architecture.

We depict a simplified three-layer architecture of our system in Figure 6.2. The bottom layer, i.e., the context information layer, consists of context data providers (CP) offering context information from various sources ranging from static information to sensor values. Thereby, different context providers may provide data with different levels of detail. In addition, this data can be based on different kinds of sensors. These two characteristics are the reason to specify the uncertainty of the data. Moreover, the fact that several context providers may provide data on the same phenomenon is the reason to specify the inconsistency of this data. Finally, in this open system, information about the trust in context providers is essential

to estimate the value of the provided data. We explain the details of these three aspects of quality in Section 6.2.2.

The middle layer, i.e., the federation layer, is the platform for processing queries on the data provided by the context information layer. Thereby, the federation nodes (FN) on this layer provide the abstraction of a single data source to the applications (App) in the application layer. Processing of data quality is done based on the currently available data at the different providers. This processing is not influenced by limitations through network characteristics, e.g., interpolation mechanisms are incorporated to cope with high network delay.

6.2.2 Three Aspects of Quality of Data

In the following discussion, o, o_1, o_2, \dots denote objects. Objects are sets of attributes. The P attribute of o_1 is denoted by $o_1.P$. Here, we only regard attributes (called P) with scalar values, representing not only, e.g., temperature or other sensor measurement values, but also – as in the following discussion – position values in a one dimensional space. This is primarily for simplicity, but, depending on the data model, can also practically be used, e.g., for representing positions of cars on a highway [dAG05].

Different data providers can manage the same object. We call the data providers $1, 2, \dots$, and $o_1^2.P$ denotes the position of object o_1 according to provider 2.

Here we assume that data providers specify a normal PDF for uncertain positions, however, due to the way we handle data of not fully reliable providers when fusing the quality aspects, we want to be able to express that, with some probability, we are not sure or do not know the value.

Definition An uncertain position P is represented by a special PDF $p : \mathbb{R} \rightarrow \mathbb{R}_0^+$ with $0 \leq \int_{-\infty}^{\infty} p(x) dx \leq 1$. With the probability $1 - \int_{-\infty}^{\infty} p(x) dx$, the value is unknown (NULL).

Besides representing uncertain positions, we also require a means for measuring how uncertain a position is. For this, we adopt the concept of differential entropy from [SW69], which was already used for measuring quality of data in [CKP03]. To be able to use this definition, we restrict the position PDF to have a lower bound l and upper bound u , with

$$p(x) \begin{cases} > 0, & l \leq x \leq u \\ = 0, & \text{otherwise} \end{cases} .$$

Definition $u(P) = - \int_l^u p(x) \log_2 p(x) dx$ is the uncertainty of position P .

This definition restricts the form of the PDF and may not be adequate for cases where the probability for the value being NULL is greater than 0, however, as shown in Section 6.2.4, we only apply this definition to values directly retrieved from data providers, where these limitations are reasonable.

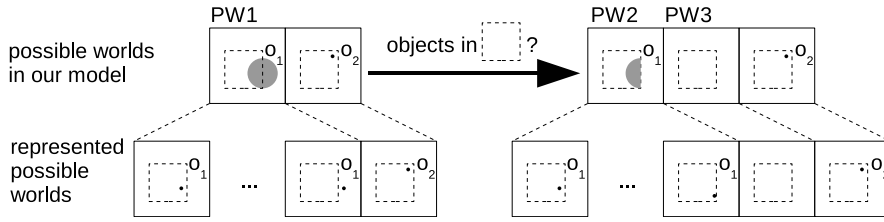


Figure 6.3: Extending the possible worlds model to support uncertainty

For measuring the consistency $c(P_1, P_2)$ of two positions P_1 and P_2 , we use the consistency metrics presented in Section 4.1.

The users and providers can have different opinions about the reliability of the provided context information and about the trustworthiness of other users and providers. It is therefore not possible to compute objective, global values for the trustworthiness of a user and the reliability of a context information object. Instead, these values are computed from the subjective point of view of the requesting user as described in chapter 5. The value $v(o_k^i.P) \in [0, 1]$ refers to the resulting reliability value of the attribute P of a context information object o_k of data provider i from the point of view of the requesting user.

6.2.3 Query Processing

We use the possible worlds approach [BGMP92, ABS⁺06, BSH⁺08, CCX08] as basis for the query processing, but need to be able to represent uncertainty, so we have to extend the model to support an infinite number of possible worlds. This is subject to ongoing research, but for queries with simple selection predicates, the approach shown in Figure 6.3 is reasonable. In addition to enumerate a finite number of possible worlds (boxes in Figure 6.3), we allow uncertain attributes in a possible world (grey circles representing positions), so that a possible world in our model can represent an infinite number of exact possible worlds (shown in the bottom part of Fig. 6.3). In contrast to the original possible world model, we need to adapt operators for our approach. Figure 6.3 shows a range query, which only a part of the o_1 s represented by $PW1$ fulfills, so the result of applying the query to $PW1$ is an empty possible world ($PW3$), and a possible world with a modified position for o_1 ($PW2$).

The Nexus system is not only able to simply retrieve objects, but can also process more complex queries. It provides a set of generic operators, which is similar to the relational algebra. The precise definition of the complete set is beyond the scope of this paper, but we briefly describe the operators used in the example scenario. Note that these operators only handle uncertainty, we explain in Section 6.2.4, why this is sufficient.

Selection σ_{pred} : The selection operator is equivalent to the selection operator of the relational algebra. It takes a list of objects as input and outputs a list

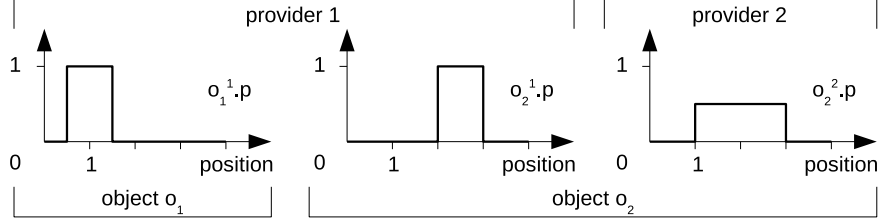


Figure 6.4: Example scenario: PDFs of the positions of o_1 and o_2

containing all objects from the input list, which fulfill the predicate $pred$. When applied to uncertain data, objects fulfill the predicate with some probability, and objects are included in the result list with this probability, i.e., σ can create several alternative results (possible worlds) and is an entity-based non-aggregate operator according to the classification in [CKP03]. As previously explained, it may be necessary to modify uncertain attribute values. NULL values are handled as in SQL: When $pred$ evaluates to *unknown*, the object is not included in the result.

Sorting $sort_{expr}$: The sorting operator sorts a list of objects. $expr$ is an expression based on attributes of an object. It is evaluated for each object in turn, and the objects are sorted according to the results. Like the selection operator, sorting can create several alternative results when applied to uncertain data. The probability of a result list is determined by the probability that evaluating $expr$ in sequence on all objects of this list results in a sorted list. Sorting is an entity-based aggregate operator.

Fetch $fetch_n$: The fetch operator just cuts a list of objects to the first n objects. It does not evaluate attributes like the other two operators do, thus does not require an adaption to handle uncertain data. We use the fetch operator in conjunction with sorting to implement a nearest neighbor query.

6.2.3.1 Example Scenario

Fig. 6.4 shows the example scenario. Two providers 1 and 2 store two objects o_1 and o_2 . For o_2 , each provider offers a representation, these two representations are different.

The uncertainties of the positions are $u(o_1^1.P) = u(o_2^1.P) = 0, u(o_2^2.P) = 1$, the consistency of $o_2.P$ is $c(o_2^1.P, o_2^2.P) = 0.75$. We want to answer the query, which of the objects located between the positions 1 and 3 is closest to position 0, more formally

$$fetch_1(sort_{dist(0,o.P)}(\sigma_{1 \leq o.P \leq 3}[o_1, o_2])) .$$

$dist$ calculates the distance between its arguments. As in this scenario, the first argument is the position 0, the result has the same PDF as the second argument.

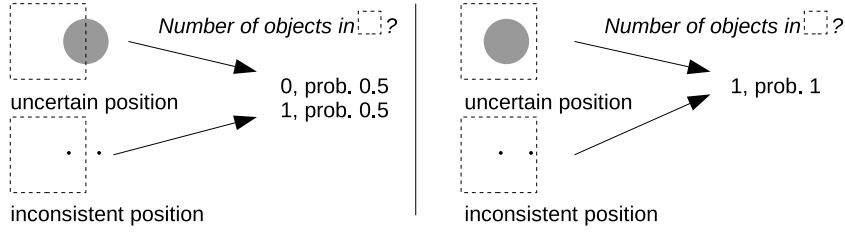


Figure 6.5: Measuring the quality of the query result

As explained above, it may be necessary to adapt the PDF for the position during selection. For a selection of the form $\sigma_{l \leq P \leq u}$, we do this by narrowing the range, where the PDF is > 0 , to the interval $[l, u]$ and multiplying the resulting function with a constant factor, so that the integral equals to 1:

$$p'(x) = \begin{cases} \frac{p(x)}{\int_l^u p(x) dx}, & l \leq x \leq u \\ 0, & \text{otherwise} \end{cases}$$

For evaluating *sort*, we must calculate the probability that a distance D_2 is greater than an other distance D_1 . When D_1 and D_2 are represented by two PDFs d_1 and d_2 , the probability for $D_2 > D_1$ is¹

$$\int_{-\infty}^{\infty} \int_{x_1}^{\infty} d_1(x_1) d_2(x_2) dx_2 dx_1 . \quad (6.1)$$

In the given scenario, we cannot be sure if o_1 actually fulfills the selection predicate, and – according to the data of provider 2 – there is a chance that o_2 is closer to 0 than o_1 . Obviously, the probability for o_1 to be closer to 0 is much higher than for o_2 . However, the probability for o_1 to fulfill the selection predicate is only 0.5, so we expect the probability for o_2 being the result of the query to be only a little bit above 0.5.

6.2.4 Processing Model

In Section 6.2.2, we presented approaches for representing and measuring uncertainty, inconsistency and trust on the data provider level. To be able to process complex queries like the one presented in the previous section, we need to address two additional questions: how to account for the quality aspects during the processing of queries and how to measure the quality of the final result set.

The straightforward attempt to solve the first problem would be to define separately for each operator, how each quality aspect is handled. When, e.g., the selection operator is applied to an uncertain attribute, the uncertainty selection operator

¹ $d_1(x_1)d_2(x_2)$ is the combined PDF for D_1 and D_2 . To derive the probability for $D_2 > D_1$, we need to integrate over the area where $x_2 > x_1$.

6 Reference Model

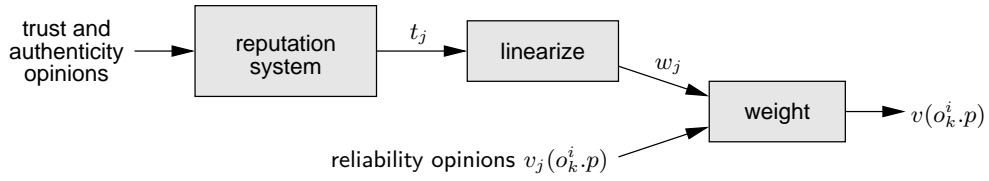


Figure 6.6: Processing model (data providers, trust and reliability)

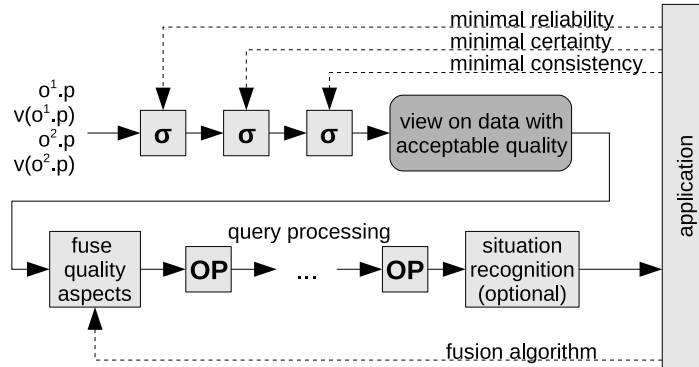


Figure 6.7: Processing model (data fusion and query processing)

would be invoked, and for an inconsistent attribute the inconsistency selection operator. However, this approach cannot handle information that is both uncertain and inconsistent, like $o_2.P$ in Figure 6.4. Therefore, we need a more integrated concept, which can deal with all three quality aspects simultaneously.

To measure the quality of result sets, in some cases, it is possible to directly apply the definitions presented in Section 6.2.2 to query results. When an object with an uncertain position is present in a result set, the uncertainty model can be used to represent its position. Likewise, the inconsistency model can be used when two different values for the same attribute of an object are in the result set. However, when only one value qualifies for the result set, the inconsistency information gets lost. To use the trust and reliability model, the reliability value of the provider can be assigned to each attribute he provides to the result. However, in more complex situations, these definitions are not suitable. Figure 6.5 shows on the left hand side a situation where we are not sure if the answer to the query *How many objects are located inside the dashed square?* is 0 or 1. This should somehow be reflected by the result's quality, but is unclear if this is uncertainty or inconsistency, because exactly the same result can be caused by uncertainty (top) or by inconsistency (bottom). On the right hand side, we have the same situation with a slightly shifted square for the query. In this case, we can be sure that the result of the query is 1, so the quality of the result should be optimal, although the data used for answering the query is uncertain or inconsistent.

To address these two problems, we are investigating the approach depicted in Figures 6.6 and 6.7. The main idea is to combine the three aspects before the actual query processing takes place, and define query processing and the result's quality based on the possible worlds model. In the following, we discuss reasons for choosing this approach.

Viewed from the perspective of query processing, uncertainty and inconsistency describe similar phenomena – there exist several alternatives for one value. In the case of uncertainty, the number of alternatives is possibly infinite, whereas in the case of inconsistency, a finite number of alternatives exist. In that sense, uncertainty is a generalization of inconsistency and both can be expressed by an uncertainty model.

When expressing inconsistency as uncertainty, we basically add all PDFs for an attribute. Thereby we have to weight the individual PDFs of the data providers, in the most simple case with the reciprocal of the number of data providers. In our case, however, we can refine the weighting using the trust and reliability values, so that PDFs from reliable data providers gain a higher weight than those from less reliable ones. This meets the supposable expectation of users that information of reliable data providers is more likely to be true.

In more detail, the approach consists of the following steps:

1. Applications or users may want to specify minimum requirements for certainty, consistency and reliability for the data used for processing the query. Three additional selections are performed before the actual query is processed which result in a subset of the original data set, that fulfills the quality constraints. Note that the selection of sufficiently reliable data providers has to be done before evaluating the consistency constraint, otherwise, unreliable providers would be able to force the removal of attributes from the subset by providing incorrect representations of the attribute, thus decreasing the consistency.
2. The three quality aspects are combined based on the uncertainty model by fusing the different representations. Typically, inconsistency is incorporated by averaging the representations, reliability by weighting them. Different applications may require different fusion algorithms, so we provide a way for the application to specify the algorithm to use.
3. The calculation of the quality of the query's result is still an open issue, but using some extension of an entropy based approach seems to be promising. It is not necessary to use an additional selection here, the application itself can decide whether the quality of the result is sufficient or not and discard the result in the latter case.

We have developed two different general fusion algorithms. The first one is highly scalable, but has the potential drawback of generating high probabilities for NULL values. When the providers $1, \dots, n$ provide values for the position of an object o ,

6 Reference Model

the resulting position is calculated as

$$o.p(x) = \frac{1}{n} \sum_{i=1}^n v(o^i.p)(o^i.p(x)) .$$

Note that $\int_{-\infty}^{\infty} o.p(x) dx$ may be smaller than 1 (cf. Section 6.2.2).

The second algorithm reduces the probability of NULL values, but its complexity grows exponentially with the number of data providers. For sake of simplicity, we show the definition for two data providers 1 and 2:

$$\begin{aligned} o.p(x) &= \frac{v(o^1.p) \cdot v(o^2.p)}{\int_{-\infty}^{\infty} o^1.p(x) \cdot o^2.p(x) dx} \cdot o^1.p(x) \cdot o^2.p(x) \\ &+ v(o^1.p) \cdot (1 - v(o^2.p)) \cdot o^1.p(x) \\ &+ (1 - v(o^1.p)) \cdot v(o^2.p) \cdot o^2.p(x) \end{aligned}$$

When $\int_{-\infty}^{\infty} o^1.p(x) \cdot o^2.p(x) dx = 0$, we omit the first summand.

An additional benefit of this approach is that the combined data quality model is closely related to models typically used in the literature, which allows us to define the semantics of our operators based on well understood concepts.

6.2.5 Revisiting the Example Scenario

In this section, we describe how the query in Section 6.2.3.1 is processed using our processing model. We use the notation $[o_1, \dots, o_n]_p$ for a result list generated with probability p .

For the first example, all objects of all data providers are considered fully reliable, i.e., $v(o_1^1.p) = v(o_2^1.p) = v(o_2^2.p) = 1$ and we do not use restrictions on certainty, consistency and reliability. Thus, fusing the data of the two providers results in $o_1 = o_1^1$ and o_2 with

$$o_2.p(x) = \begin{cases} 0.25, & 1 \leq x < 2 \\ 0.75, & 2 \leq x \leq 3 \\ 0, & \text{otherwise} \end{cases} .$$

Figure 6.8 shows the intermediate results after each operator of the query and the final result. σ does not modify o_2 , because its position lies completely inside the requested area, o_1 , however, becomes o_1' with

$$o_1'.p(x) = \begin{cases} 2, & 1 \leq x \leq 1.5 \\ 0, & \text{otherwise} \end{cases} .$$

o_1' is closer to 0 than o_2 with a probability of $\frac{15}{16}$ according to (6.1). The probability of o_2 being the final result of the query is a little bit higher than 0.5 as expected in Section 6.2.3.1.

6.2 Integrated Quality-Aware Processing Model

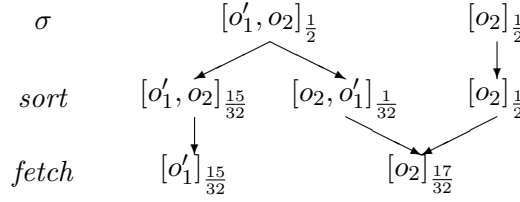


Figure 6.8: Processing the query ($v(o_1^1.p) = v(o_2^1.p) = v(o_2^2.p) = 1$)

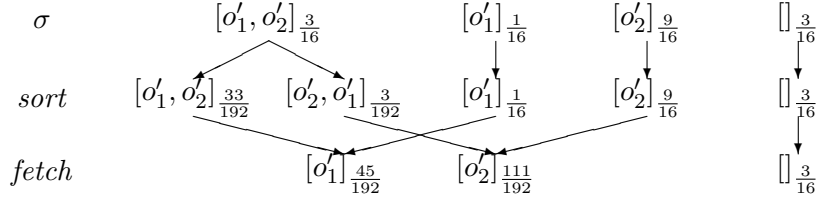


Figure 6.9: Processing the query ($v(o_1^1.p) = v(o_2^1.p) = 0.5$, $v(o_2^2.p) = 1$)

For the second example, shown in Figure 6.9, we use the reliability values $v(o_1^1.p) = v(o_2^1.p) = 0.5$ and $v(o_2^2.p) = 1$. This results in the following situation after fusing the data:

$$o_1.p(x) = \begin{cases} 0.5, & 0.5 \leq x \leq 1.5 \\ 0, & \text{otherwise} \end{cases} \quad o_2.p(x) = \begin{cases} 0.25, & 1 \leq x < 2 \\ 0.5, & 2 \leq x \leq 3 \\ 0, & \text{otherwise} \end{cases} .$$

Because we do not have full confidence in the reliability of provider 1, $o_1.P$ is NULL with probability 0.5 and $o_2.P$ with probability 0.25. o_1 fulfills the selection predicate with a probability of 0.25, o_2 with a probability of 0.75, so the selection also modifies $o_2.p$:

$$o'_1.p(x) = \begin{cases} 2, & 1 \leq x \leq 1.5 \\ 0, & \text{otherwise} \end{cases} \quad o'_2.p(x) = \begin{cases} \frac{1}{3}, & 1 \leq x < 2 \\ \frac{2}{3}, & 2 \leq x \leq 3 \\ 0, & \text{otherwise} \end{cases} .$$

Equation (6.1) yields $\frac{11}{12}$ for the probability of o'_1 being closer to 0 than o'_2 .

7 Conclusions

This report presented an overview of the developed reference model for *Quality of Context Information* (QoC). QoC comprises numerous aspects, such as the accuracy of sensed data, transmission and update protocols, consistency between data from several providers, or the trust placed into the information from individual providers. Our *QoC reference model* enables a federated context management system such as Nexus to incorporate QoC in an integrated manner, from the sensors to the applications. The model was introduced in five main parts: Abstract framework for quality aspects, Degradation model, Consistency model, Trust model and Integrated QoC-aware processing model.

Abstract framework for quality aspects: We distinguish three abstraction levels of context information (sensor information, observable context, and high-level context). To satisfy all requirements, our proposed abstract framework distinguishes three fundamental quality aspects: degradation, consistency, and trust. Each of these aspects has to be modeled along the layers of the system: low level sensor data, context information layer and high level context. This leads to the presented 3x3 framework of quality aspects which are only at the first glance rather independent.

Degradation model: We proposed an integrated approach how to cope with physical limitations of sensors and the corresponding sensing inaccuracies, imprecision, etc.—from the raw sensor data via observable context to high-level context. Moreover, for applications, we presented QoC-aware interfaces for querying context information. One main aspect in a large-scale context-aware systems is the need for a generic model for position information. We discussed the requirements for a suitable model including an uncertainty-aware query interface. Our survey showed that existing technology-specific uncertainty models can be considered as partial spatial distribution functions (psdf). We proposed an extended query interface for position information by extending common query types with information on the position uncertainty. This approach can be extended to scalar types as well as data with three or more dimensions. For high level contexts we presented surveys and requirements for degradation models in situation detection.

Consistency model: Inconsistency in large-scale context-aware systems can be caused by different data providers offering different values for the same attribute. The basic design idea of our consistency model is to estimate the probability that the representation of one provider does not conflict with the representation of the other provider. Our model depends on the types of the attributes and distinguishes between four domains: discrete domain, continuous domain with and without pdf and the 3D-domain. For each domain inconsistency can be observed on the context information layer and on the high-level context layer.

7 Conclusions

Trust model: With an open context-management platform such as Nexus, trustworthiness of context providers is an important issue. For this reason we propose a comprehensive approach for assessing the reliability and trustworthiness of each provider. Our detailed model represents trust and authenticity statements as well as trust values. These values allow to express degrees of belief, ignorance and disbelief. The computation of the resulting trust is based on a probability theoretical model.

Integrated QoC-aware processing model: We proposed a universal QoC-aware processing model for queries for context information of arbitrary applications. This model incorporates the specific models for degradation, consistency, and trust and the dependencies between these aspects.

In the future, we are looking to extend our work for other types of context information. This comprises three main parts: 1) an extension to degraded image data and the higher level context derived from it; 2) the handling of degraded trajectories, either from sensor inaccuracies or artificially introduced uncertainties for privacy reasons; 3) a description of the quality of partial models, especially focusing on the completeness of the model with regard to the real world.

Bibliography

- [ABS⁺06] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha U. Nabar, Tomoe Sugihara, and Jennifer Widom. Trio: A system for data, uncertainty, and lineage. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *VLDB*, pages 1151–1154. ACM, 2006.
- [ACGW99] John Michael Ashley, Matthew Copeland, Joergen Grahn, and David A. Wheeler. *The GNU Privacy Handbook*. The Free Software Foundation, 1999.
- [AFGS08] D. Akca, M. Freeman, A. Gruen, and I. Sargent. Quality assessment of 3D building data by 3D surface matching. In *21th ISPRS Congress. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXVII, part B2*, pages 771–777, Beijing, China, July 2008.
- [Bac96] Johann Bacher. *Clusteranalyse, Anwendungsorientierte Einführung*. Oldenbourg, 2nd edition, 1996. in German.
- [Bal87] J. F. Baldwin. Evidential Support Logic Programming. *Fuzzy Sets Systems*, 24(1):1–26, 1987.
- [BDR07] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, 2(4):263–277, June 2007.
- [Bel75] Nuel D. Belnap. A Useful Four-valued Logic. In *Modern Uses of Multi-valued Logic*, pages 8–37, 1975.
- [BGMP92] Daniel Barbará, Hector Garcia-Molina, and Daryl Porter. The management of probabilistic data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, 1992.
- [BKZ⁺08] Ruben Benkmann, Uwe-Philipp Käppeler, Oliver Zweigle, Reinhard Lafrenz, and Paul Levi. Resolving Inconsistencies using Multi-agent Sensor Systems. In Luis Seabra Lopez, Filipe Silva, and Vitor Santos, editors, *Proceedings of the 8th Conference on Autonomous Robot Systems and Competition: Robotica 08*, pages 93–98, Aveiro, April 2008. Universidade de Aveiro.

Bibliography

- [BKZ⁺09] Ruben Benkmann, Uwe-Philipp Käppeler, Oliver Zweigle, Reinhard Lafrenz, and Paul Levi. Resolving Inconsistencies Using Multi-Agent Sensor Systems. *Robotica*, 03/09(76):22–27, November 2009.
- [BMK⁺00] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. EasyLiving: Technologies for intelligent environments. *Lecture notes in computer science*, pages 12–29, 2000.
- [BP00] Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proc. of 19th INFOCOM*, pages 775–784, March 2000.
- [BSH⁺08] Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, Martin Theobald, and Jennifer Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.
- [Bun94] WL Buntine. Operations for learning with graphical models. *Arxiv preprint cs/9412102*, 1994.
- [CCX08] Reynold Cheng, Jinchuan Chen, and Xike Xie. Cleaning uncertain data with quality guarantees. *PVLDB*, 1(1):722–735, 2008.
- [CFJ04] H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(03):197–207, 2004.
- [CG96] Anthony G. Cohn and Nicholas M. Gotts. The ‘egg-yolk’ representation of regions with indeterminate boundaries. In *Proceedings GISDATA Specialist Meeting on Spatial Objects with Undetermined Boundaries*, pages 171–187, 1996.
- [CKP03] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In Alon Y. Halevy, Zachary G. Ives, and AnHai Doan, editors, *SIGMOD Conference*, pages 551–562. ACM, 2003.
- [CKP04] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving object environments. *IEEE Trans. on Know. and Data Eng.*, 16(9):1112–1127, September 2004.
- [DA00] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *Proc. of CHI 2000 Workshop on the What, Who, Where, When and How of Context-Awareness*, The Hague, Netherlands, April 2000.
- [dAG05] Victor Teixeira de Almeida and Ralf Hartmut Güting. Supporting uncertainty in moving objects in network databases. In Cyrus Shahabi and Omar Boucelma, editors, *GIS*, pages 31–40. ACM, 2005.

- [Dem04] Robert Demolombe. Reasoning About Trust: A Formal Logical Framework. In *Proceedings of the Second International Conference of Trust Management (iTrust 2004)*, pages 291–303, 2004.
- [Die91] C. F. Dietrich. *Uncertainty, Calibration and Probability: The Statistics of Scientific and Industrial Measurement*. Adam Hilger, 2nd edition, 1991.
- [DKN⁺06] Dominique Dudkowski, Uwe-Philipp Käppeler, Daniela Nicklas, Thomas Schwarz, Oliver Siemoneit, Steffen Volz, Klaus Wieglerling, and Oliver Zweigle. Konsistenz in Nexus. Technical Report 2006/11, Universität Stuttgart: SFB 627 (Nexus: Umgebungsmodelle für mobile kontextbezogene Systeme), März 2006.
- [DKW87] J. De Kleer and B.C. Williams. Diagnosing multiple faults. *ARTIFICIAL INTELLIG.*, 32(1):97–130, 1987.
- [DLR⁺77] A.P. Dempster, N.M. Laird, D.B. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.
- [EC95] Timothy C. Earle and George T. Cvetkovich. *Social Trust. Toward a Cosmopolitan Society*. Praeger/Greenwood, Westport, 1995.
- [Edw90] D. Edwards. Hierarchical interaction models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 52(1):3–20, 1990.
- [FC04] Patrick Fahy and Siobhán Clarke. Cass - middleware for mobile context-aware applications. In *Workshop on Context Awareness at MobiSys 2004*, Boston, MA, USA, June 2004.
- [FHL⁺03] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE Per. Comp.*, 2(3):24–33, July 2003.
- [fMuG95] Internationales Büro für Maß und Gewicht. Guide to the expression of uncertainty in measurement, din 01319. ISO/BIPM-Leitfaden, 1995.
- [För92] W. Förstner. *Robust Computer Vision: Quality of Vision Algorithms*, chapter Uncertainty of geometric entities, pages 30–44. Wichmann Verlag, Karlsruhe, Germany, 1992.
- [Fre92] Christian Freksa. Using orientation information for qualitative spatial reasoning. In A. U. Frank, I. Campari, and U. Formentini, editors, *International Conference on Theories and Methodes of Spatio-Temporal Reasoning in Geographic Space*, volume 639, pages 114–124, Pisa, Italy, 1992. Springer.

Bibliography

- [FS08] M. Freeman and I. Sargent. Quantifying and visualising the uncertainty in 3D building model walls using terrestrial lidar data. In *Proceedings of the Remote Sensing and Photogrammetry Society Conference 2008 'Measuring change in the Earth system'*, Exeter, UK, September 2008.
- [Gam88] Diego Gambetta. *Can We Trust Trust?*, pages 213–237. Basil Blackwell, 1988. Reprinted in electronic edition from Dept. of Sociology, University of Oxford, Chapter 13.
- [GHS08] Andreas Gutscher, Jessica Heesen, and Oliver Siemoneit. Possibilities and Limitations of Modeling Trust and Reputation. In Manuel Möller, Thomas Roth-Berghofer, and Wolfgang Neuser, editors, *Proceedings of the Fifth International Workshop on Philosophy and Informatics WSPI 2008*, volume 332 of *CEUR Workshop Proceedings*. CEUR-WS.org, April 2008.
- [Gle01] Michael Glemser. Zur Berücksichtigung der geometrischen Objektunsicherheit in der Geoinformatik. Deutsche Geodätische Kommission, Reihe C, Nr. 539, München, 2001.
- [GPZ04] T. Gu, H.K. Pung, and DQ Zhang. A bayesian approach for dealing with uncertain contexts. *Advances in Pervasive Computing*, pages 136–144, 2004.
- [GS00] Tyrone Grandison and Morris Sloman. A Survey of Trust in Internet Application. *IEEE Communications Surveys & Tutorials*, 3(4):2–16, 2000.
- [GS05] Ralf Hartmut Güting and Markus Schneider. *Moving Objects Databases*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2005.
- [Gut07] Andreas Gutscher. A Trust Model for an Open, Decentralized Reputation System. In *Proceedings of the Joint iTrust and PST Conferences on Privacy Trust Management and Security (IFIPTM 2007)*, pages 285–300, 2007.
- [Gut08] Andreas Gutscher. Reasoning with Uncertain and Conflicting Opinions in Open Reputation Systems. In *Proceedings of the Fourth International Workshop on Security and Trust Management (STM 2008)*, 2008.
- [Gut09] Andreas Gutscher. A Method to Evaluate Uncertain and Conflicting Trust and Authenticity Statements. In *Proceedings of the 1st International Workshop on Managing Insider Security Threats (MIST 2009)*, pages 62–82, 2009.
- [Haa96] Norbert Haala. Gebäuderekonstruktion durch Kombination von Bild- und Höhendaten. Deutsche Geodätische Kommission, Reihe C, Nr. 460, München, 1996.

- [HAES08] M.A. Hossain, P.K. Atrey, and A. El Saddik. Learning Multi-Sensor Confidence using Difference of Opinions. In *IEEE Intl. Instrumentation & Measurement Technology Conf*, 2008.
- [HB01] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, August 2001.
- [HCD04] Farookh Khadeer Hussain, Elizabeth Chang, and Tharam S. Dillon. Classification of trust in Peer-to-Peer (P2P) communication. *Computer Systems: Science & Engineering*, 19(2), 2004.
- [HH94] Andy Harter and Andy Hopper. A distributed location system for the active office. *IEEE Network*, 8(1):62–70, January 1994.
- [HIMB05] Karen Henriksen, Jadwiga Indulska, Ted McFadden, and Sasitharan Balasubramaniam. Middleware for distributed context-aware systems. In *Proceedings of the OTM Confederated International Conferences, CoopIS, DOA, and ODBASE*, pages 846–863, Agia Napa, Cyprus, October 2005.
- [HIR02] K. Henriksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. *Lecture notes in computer science*, pages 167–180, 2002.
- [HKBE07] Andreas Hub, Stefan Kombrink, Klaus Bosse, and Thomas Ertl. Tania – a tactile-acoustical navigation and information assistant for the 2007 csun conference. In *Proc. of 22nd CSUN*, Los Angeles, CA, USA, March 2007.
- [HKL00] Rolf Haenni, Jürg Kohlas, and Norbert Lehmann. *Probabilistic Argumentation Systems*, volume 5 (Algorithms for Uncertainty and Defeasible Reasoning) of *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 221–288. Springer, 2000.
- [HKN⁺05] Nicola Hönle, Uwe-Philipp Käppeler, Daniela Nicklas, Thomas Schwarz, and Matthias Großmann. Benefits of integrating meta data into a context model. In *PerCom Workshops*, pages 25–29, 2005.
- [HL04] David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression: Textbook and Solutions Manual*. Wiley-IEEE, 2nd edition, 2004.
- [Hou94] A. Hou. A theory of measurement in diagnosis from first principles. *Artificial Intelligence*, 65(2):281–328, 1994.
- [HS07] Christoph Hubig and Oliver Siemoneit. Vertrauen und Glaubwürdigkeit in der Unternehmenskommunikation. In Ansgar Piwinger, Manfred; Zerfaß, editor, *Handbuch Unternehmenskommunikation*. Gabler, 2007.

Bibliography

- [HS09] Christoph Hubig and Oliver Siemoneit. Vertrauen und Glaubwürdigkeit als kommunikationspolitische Ziele erfolgreicher IR. In Manfred Piwinger, editor, *Praxishandbuch Investor Relations*. Gabler, 2009.
- [HSP⁺03] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. Context-awareness on mobile devices – the hydrogen approach. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS 2003)*, pages 10–19, January 2003.
- [JGK06] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Simplification and Analysis of Transitive Trust Networks. In *Web Intelligence and Agent Systems Journal*, pages 139–161, 2006.
- [JH05] Jacek Jonczyk and Rolf Haenni. Credential Networks: a General Model for Distributed Trust and Authenticity Management. In *PST*, pages 101–112, 2005.
- [JI02] Audun Jøsang and Roslan Ismail. The beta reputation system. In *Proceedings of the 15th Bled Conference on Electronic Commerce*, 2002.
- [JIB07] Audun Jøsang, Roslan Ismail, and Colin Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. In *Decision Support Systems*, 2007.
- [Jøs97] Audun Jøsang. Artificial Reasoning with Subjective Logic. In *Proceedings of the Second Australian Workshop on Commonsense Reasoning*, 1997.
- [JP95] R. Jiroušek and S. Přeučil. On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics & Data Analysis*, 19(2):189, 1995.
- [Kad07] Martin Kada. Scale-dependent simplification of 3d building models based on cell decomposition and primitive instancing. In Stephan Winter, Matt Duckham, Lars Kulik, and Benjamin Kuipers, editors, *COSIT*, volume 4736 of *Lecture Notes in Computer Science*, pages 222–237. Springer, 2007.
- [Käp08] Uwe-Philipp Käppeler. Modellierung der Degradierung von skalaren Sensordaten in ContextServer und SensorContextServer. Technical Report 2008/01, Universität Stuttgart: SFB 627 (Nexus: Umgebungsmodelle für mobile kontextbezogene Systeme), Oktober 2008.
- [Käp09] Uwe-Philipp Käppeler. Metriken für Qualitätskennzahlen zur Degradierung von skalaren Sensordaten. Technical Report 2009/02, Universität Stuttgart: SFB 627 (Nexus: Umgebungsmodelle für mobile kontextbezogene Systeme), November 2009.

- [KBZ⁺08] Uwe-Philipp Käppeler, Ruben Benkmann, Oliver Zweigle, Reinhard Lafrenz, and Paul Levi. Resolving Inconsistencies in Shared Context Models using Multiagent Systems. In Rüdiger Dillmann and Wolfram Burgard, editors, *Proceedings of the 10th International Conference on Intelligent Autonomous Systems IAS-10*, pages 93–98, Baden Baden, July 2008. Springer-Verlag.
- [KGS⁺09] Uwe-Philipp Käppeler, Andreas Gerhardt, Christian Schieberle, Matthias Wiselka, Kai Häussermann, Oliver Zweigle, and Paul Levi. Reliable Situation Recognition based on Noise Levels. In K. Duncan and C. A. Brebbia, editors, *Proceedings of the First International Conference on Disaster Management and Human Health Risk*, volume 110 of *WIT Transactions on the Built Environment*, pages 127–137, New Forest, UK, September 2009. WIT Press.
- [KMK⁺03] Panu Korpipää, Jani Mäntyjärvi, Juha Kela, Heikki Keränen, and Esko-Juhani Malm. Managing context information in mobile devices. *IEEE Pervasive Computing*, 2(3):42–51, July 2003.
- [Koh07] Reto Kohlas. *Decentralized Trust Evaluation and Public-Key Authentication*. PhD thesis, Universität Bern, 2007.
- [KR95] R.E. Kass and A.E. Raftery. Bayes factors and model uncertainty. *Journal of the American Statistical Association*, 90(773):95, 1995.
- [KR03] Michael Kinateder and Kurt Rothermel. Architecture and Algorithms for a Distributed Reputation System. In *Proceedings of the First International Conference on Trust Management (iTrust 2003)*, volume 2692 of *LNCS*, pages 1–16, 2003.
- [KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th International Conference on World Wide Web*, pages 640–651, 2003.
- [Lau95] S.L. Lauritzen. *The EM algorithm for graphical association models with missing data*. The University of Aalborg, Institute for Electronic Systems, Department of Mathematics and Computer Science, 1995.
- [LCG⁺09a] Ralph Lange, Nazario Cipriani, Lars Geiger, Matthias Großmann, Harald Weinschrott, Andreas Brodt, Matthias Wieland, Stamatia Rizou, and Kurt Rothermel. Making the world wide space happen: New challenges for the Nexus context platform. In *PerCom Workshops*. IEEE Computer Society, 2009. (to appear).
- [LCG⁺09b] Ralph Lange, Nazario Cipriani, Lars Geiger, Matthias Großmann, Harald Weinschrott, Andreas Brodt, Matthias Wieland, Stamatia Rizou,

Bibliography

- and Kurt Rothermel. Making the world wide space happen: New challenges for the nexus context platform. In *Proc. of 7th PerCom*, pages 300–303, Galveston, TX, USA, March 2009.
- [LDR08] Ralph Lange, Frank Dürr, and Kurt Rothermel. Online trajectory data reduction using connection-preserving dead reckoning. In *Proc. of 5th MobiQuitous*, Dublin, Ireland, July 2008.
- [LFDR09] Ralph Lange, Tobias Farrell, Frank Dürr, and Kurt Rothermel. Remote real-time trajectory simplification. In *Proc. of 7th PerCom*, Galveston, TX, USA, March 2009.
- [LR01] Alexander Leonhardi and Kurt Rothermel. A comparison of protocols for updating location information. *Cluster Computing: The Journal of Networks, Software Tools and Applications*, 4(4):355–367, October 2001.
- [Luh79] Niklas Luhmann. Trust: A Mechanism for the Reduction of Social Complexity. In *Trust and Power: Two Works by Niklas Luhmann*. Wiley and Sons, 1979.
- [Mar94] Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.
- [Mau96] Ueli Maurer. Modelling a Public-Key Infrastructure. In E. Bertino, editor, *Proc. 1996 European Symposium on Research in Computer Security (ESORICS' 96)*, volume 1146 of *Lecture Notes in Computer Science*, pages 325–350. Springer-Verlag, 1996.
- [MD05] Stephen Marsh and Mark R. Dibben. Trust, Untrust, Distrust and Mistrust – An Exploration of the Dark(er) Side. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of Third iTrust International Conference (iTrust 2005), Paris, France, May 23-26, 2005*, volume 3477, pages 17–33. Springer, May 2005.
- [Men01] Scott William Menard. *Applied Logistic Regression Analysis*. Sage Publications, Inc, 2nd edition, October 2001.
- [Mie03] Andrea Miene. *Räumlich-zeitliche Analyse von dynamischen Szenen*. PhD thesis, Universität Bremen, 2003.
- [MKT08] Muhammad Atif Mehmood, Lars Kulik, and Egemen Tanin. Autonomous navigation of mobile agents using rfid-enabled space partitions. In *Proc. of 16th ACM GIS*, Irvine, CA, USA, November 2008.

- [MLD99] J.W. Myers, K.B. Laskey, and K.A. DeJong. Learning bayesian networks from incomplete data using evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 458–465. Citeseer, 1999.
- [MRF04] R. Mayrhofer, H. Radi, and A. Ferscha. Recognizing and predicting context by learning from user behavior. *Radiomatics: Journal of Communication Engineering, special issue on Advances in Mobile Multimedia*, 1(1):30–42, 2004.
- [MYCD] S. McKeever, J. Ye, L. Coyle, and S. Dobson. A context quality model to support transparent reasoning with uncertain context. *Proc of QuaConn Stuttgart*.
- [Nag92] Nico J. D. Nagelkerke. *Maximum Likelihood Estimation of Functional Relationships (Lecture Notes in Statistics)*. Springer Verlag, January 1992.
- [Nat02] National Marine Electronics Association. Nmea 0183 v 3.01. standard, January 2002.
- [Nea92] RM Neal. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113, 1992.
- [NGS⁺01] Daniela Nicklas, Matthias Großmann, Thomas Schwarz, Steffen Volz, and Bernhard Mitschang. A model-based, open architecture for mobile, spatially aware applications. In Christian S. Jensen, Markus Schneider, Bernhard Seeger, and Vassilis J. Tsotras, editors, *Proceedings of the 7th International Symposium on Spatial and Temporal Databases: SSTD 2001; Redondo Beach, CA, USA, July 12-15, 2001*, volume 2121 of *Lecture Notes in Computer Science*, pages 117–135. Springer-Verlag, Juli 2001.
- [OA00] Robert J. Orr and Gregory D. Abowd. The smart floor: A mechanism for natural user identification and tracking. In *Extended Abstracts on Human factors in Computing Systems (CHI 2000)*, pages 275–276, The Hague, The Netherlands, April 2000.
- [Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- [Pet09] Michael Peter. Presentation and evaluation of inconsistencies in multiply represented 3d building models. In Kurt Rothermel, Dieter Fritsch, Wolfgang Blochinger, and Frank Dürr, editors, *QuaCon*, volume 5786 of *Lecture Notes in Computer Science*, pages 156–163. Springer, 2009.

Bibliography

- [PJ99] Dieter Pfoser and Christian S. Jensen. Capturing the uncertainty of moving-object representations. In *Proc. of 6th SSD*, pages 111–131, Hong Kong, China, July 1999.
- [PL03] S. Padó and M. Lapata. Constructing semantic space models from parsed corpora. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, ACL*, volume 3, 2003.
- [PS05] Alejandro Pauly and Markus Schneider. Topological predicates between vague spatial objects. In *Proc. of 9th SSTD*, pages 418–432, Angra dos Reis, Brazil, August 2005.
- [RAMC04] A. Ranganathan, J. Al-Muhtadi, and R.H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, pages 62–70, 2004.
- [Rei87] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
- [RM96] S. Ramachandran and R.J. Mooney. Revising Bayesian network parameters using backpropagation. In *International Conference on Neural Networks: Plenary, Panel and Special Sessions*, page 82. Citeseer, 1996.
- [Rot07] Jörg Roth. Inferring position knowledge from location predicates. In *Proc. of 3rd LoCA*, pages 245–262, Oberpfaffenhofen, Germany, September 2007.
- [SBG99] Albrecht Schmidt, Michael Beigl, and Hans-Werner Gellersen. There is more to context than location. *Computers & Graphics Journal*, 23(6):893–902, December 1999.
- [SBGP04] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Priyantha. Tracking moving devices with the cricket location system. In *Proc. of 2nd MobiSys*, pages 190–202, June 2004.
- [Sch08] Markus Schneider. *Handbook of Research on Fuzzy Information Processing in Databases*, chapter Fuzzy Spatial Data Types for Spatial Uncertainty Management in Databases, pages 490–515. May 2008.
- [SF02] Kari Sentz and Scott Ferson. *Combination of Evidence in Dempster-Shafer Theory*, 2002.
- [Sha76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton Univ. Press, 1976.
- [SHF07] I. Sargent, J. Harding, and M. Freeman. Data quality in 3D: Gauging quality measures from users’ requirements. In *5th International Symposium on Spatial Data Quality*, Enschede, The Netherlands, 2007.

- [SJJ96] L.K. Saul, T. Jaakkola, and M.I. Jordan. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4(4):61–76, 1996.
- [SW69] Claude E. Shannon and Warren Weaver. *The mathematical theory of communication*. The University of Illinois Press, fourth edition, 1969.
- [Tan93] M.A. Tanner. *Tools for statistical inference*. Springer New York, 1993.
- [Tim02] Heiko Timm. *Fuzzy-Clusteranalyse: Methoden zur Exploration von Daten mit fehlenden Werten sowie klassifizierten Daten*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 6 2002.
- [TLS01] F. Tian, Y. Lu, and C. Shi. Learning Bayesian networks with hidden variables using the combination of EM and evolutionary algorithms. *Lecture notes in computer science*, pages 568–574, 2001.
- [TWZC02] Goce Trajcevski, Ouri Wolfson, Fengli Zhang, and Sam Chamberlain. The geometry of uncertainty in moving objects databases. In *Proc. of 8th EDBT*, volume 2287 of *Lecture Notes in Computer Science*, pages 233–250, March 2002.
- [UN08] United States Department of Defense and Navstar GPS. *Global Positioning System Standard Positioning Service Performance Standard*. 4th edition, September 2008.
- [vJP05] Alminas Čivilis, Christian S. Jensen, and Stardas Pakalnis. Techniques for efficient road-network-based tracking of moving objects. *IEEE Trans. on Know. and Data Eng.*, 17(5):698–712, May 2005.
- [WDC⁺04] Xiaohang Wang, Jin Song Dong, ChungYau Chin, Sanka, Ravipriya Hettiarachchi, and Daqing Zhang. Semantic space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, 3(3):32–39, July 2004.
- [WZGP04] X.H. Wang, D.Q. Zhang, T. Gu, and H.K. Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of the second IEEE annual conference on pervasive computing and communications workshops*, volume 18. IEEE Computer Society Washington, DC, USA, 2004.
- [Yag87] Ronald R. Yager. On the Dempster-Shafer Framework and New Combination Rules. *Information Sciences*, 41(2):93–137, 1987.
- [YM03] Xingbo Yu and Sharad Mehrotra. Capturing uncertainty in spatial queries over imprecise data. In *Proc. of 14th DEXA*, pages 192–201, Prague, Czech Republic, September 2003.
- [Zad84] Lotfi A. Zadeh. Review of Books: A Mathematical Theory of Evidence. *The AI Magazine*, 5(3):81–83, 1984.

Bibliography

- [ZHKL09] Oliver Zweigle, Kai Häussermann, Uwe-Philipp Käppeler, and Paul Levi. Extended TA Algorithm for adapting a Situation Ontology. In *Proceedings of the FIRA RoboWorld Congress, Progress in Robotics*, volume 44 of *Communications in Computer and Information Science*, pages 364–371, Incheon, Korea, August 2009. Springer Verlag.
- [ZKLL06] Oliver Zweigle, Uwe-Philipp Käppeler, Reinhard Lafrenz, and Paul Levi. Situation recognition for reactive agent behavior. In *Artificial Intelligence and Soft Computing*, Palma de Mallorca, August 2006. IASTED.