Department of Computer Science

Stefan Berthold

Linkability of communication contents

Keeping track of disclosed data using Formal Concept Analysis

Computer Science
D-level thesis (20p)

Date: 06-08-23
Supervisor: Simone Fischer-Hübner
Examiner: Donald F. Ross
Serial Number: D2006:07

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Linkability of communication contents

Keeping track of disclosed data using Formal Concept Analysis

Stefan Berthold

All material in this thesis which is not my own work has been identified and no material is included for which a degree has previously been conferred.

_____

Stefan Berthold

Approved, August 23, 2006

_____

Opponent: Dr. Thijs Holleboom

_____

Advisor: Prof. Simone Fischer-Hübner

_____

Examiner: Dr. Donald Ross

Acknowledgments

I would like to express my gratitude to Simone Fischer-Hübner, my supervisor at Karlstad University, who invited me to Sweden and organized my stay in a very inspiring research environment. The discussions with members of the privacy and security group in Karlstad, but in particular with Prof. Fischer-Hübner, led to valuable contributions to this thesis.

I am also very grateful for the constant and reliable support of Sebastian Clauß, my supervisor at TU-Dresden, who patiently read and annotated every part of this thesis. He encouraged me with a lot of useful suggestions to focus and continue, even in confusing situations.

In addition, I would like thank Leonardo Martucci who made friends with me just when I arrived in Karlstad and started to give me paddling lessons for free. There seemed to be even not a single moment when he was not able to come up with a funny discussion about a profound topic or the other way around. Particularly, I am happy that he did not knock me over into the water, though.

However, in particular, I would like to thank my parents who came all the way from Germany up to Sweden to bring my kitchen equipment and some electrical device after I spent a month in tents, before. Their steady support was definitely the foundation for my studies abroad and, therefore, for this work.

Abstract

A person who is communication about (the data subject) has to keep track of all of his revealed data in order to protect his right of informational self-determination. This is important when data is going to be processed in an automatic manner and, in particular, in case of automatic inquiries. A data subject should, therefore, be enabled to recognize useful decisions with respect to data disclosure, only by using data which is available to him.

For the scope of this thesis, we assume that a data subject is able to protect his communication contents and the corresponding communication context against a third party by using end-to-end encryption and Mix cascades. The objective is to develop a model for analyzing the linkability of communication contents by using Formal Concept Analysis. In contrast to previous work, only the knowledge of a data subject is used for this analysis instead of a global view on the entire communication contents and context.

As a first step, the relation between disclosed data is explored. It is shown how data can be grouped by types and data implications can be represented. As a second step, behavior, i. e. actions and reactions, of the data subject and his communication partners is included in this analysis in order to find critical data sets which can be used to identify the data subject.

Typical examples are used to verify this analysis, followed by a conclusion about pros and cons of this method for anonymity and linkability measurement. Results can be used, later on, in order to develop a similarity measure for human-computer interfaces.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Unlinkability is a requirement for anonymity, whereas the ability to perform actions anonymously plays an important role for privacy-enhancing technology. Privacy-enhancing identity management systems need, therefore, to provide reliable information to their users about links which they will cause by their actions and, particularly, by their communication.

1.1 Objective

In this thesis, we suggest a general way to keep track of disclosed data. We describe how data structures and methods of Formal Concept Analysis can be applied to communication data to show correlations and, therefore, links between items which seem to be naturally unlinkable, such as different pseudonyms, for instance.

1.2 Motivation

In many situations, anonymity is widely accepted. For instance, customers are usually accepted while browsing through a shop, even though they do not prove their identity in

the doorway. In order to keep this anonymity, it is, then, necessary to avoid revealing any information which might provide a link to their identity or to a certain part of identity. In this thesis, we take a closer look at communication contexts, i. e. messages between two parties. These messages can be part of an ordinary conversation between two persons, but might also be mails or e-mails, for instance. In particular, the latter ones provide the opportunity of standardized message forms. These forms are useful in order to avoid specific data requests at the stage of inquiry, in case such requests are not permitted by law. The Swedish government, for instance, is discussing anonymous job applications. The report [Lin05] summarizes investigations which point out that persons with foreign background are discriminated to a certain degree, even if they have the same or better skills. Other properties which have been found to influence significantly application evaluations are, for instance, name, sex, and age of an applicant. The name, however, influences evaluation results, according to [Lin05], just because it allows to conclude about the origin of an applicant. Therefore, as a matter of fact, such properties, i. e. personal data, or data items in general, need not necessarily to be independent from each other. In fact, there are relations between different data items which can lead to a more thorough identification than it would have been expected by looking at all transmitted data items only. These relations or links have to be taken in account as well as their linkability to persons, if anonymity shall be preserved with respect to specific data items, such as origin, sex, and age. An applicant, however, needs to be addressable, since the aim of an application is still to be accepted for the position or to be invited to an interview. Careful considerations are necessary to choose data items which provide anonymity with respect to several data items and also linkability with respect to a return address.

1.3 Scope

In this thesis, we focus on unlinkability and linkability, respectively, between messages and subjects caused by data items. We discuss previous approaches of measuring anonymity and linkability in the following chapter. In contrast to these previously suggested measurements which are based on probabilities and information theory, we base our approach on data items within a communication context.

Results of this thesis alone do not lead to a privacy-enhancing identity management system, but are meant to contribute to such a system. In particular, we do not suggest any privacy-enhancing mechanism, but a procedure for a more concise representation of disclosed data. This representation can, however, be used to draw conclusions about linkability of subjects, pseudonyms, and messages, in case there is enough information. The final privacy risk estimation with respect to the disclosed data items has, then, to be done separately.

Our application of Formal Concept Analysis to communication context does yet not include actions and reactions of data subjects and communication partners, as it has been demanded in the task description. We discuss the reasons for this in Section 3.6 on page 64.

1.4 Organization

In Chapter 2, we describe the key methods of Formal Concept Analysis, give an overview to important concepts and measures of privacy and anonymity, and introduce the field of privacy enhancing technology.

In Chapter 3, we show a general way to apply data structures and methods of Formal Concept Analysis to communication contexts. We begin with a broad analysis and develop the level of detail successively. We provide corresponding examples to most of the application steps.

In Chapter 4, we use the methods provided in Chapter 3 and show their use and usage

by constructing a more complex scenario.

In Chapter 5, we reason for time complexity of the previously used methods of Formal Concept Analysis and introduce the main important parts of our reference implementation.

In Chapter 6, we draw the conclusions and discuss advantages and disadvantages of our approach.

In Appendix A, we provide the complete source code of our reference implementation.

In Appendix B, we provide source code representations of several examples from this thesis ready to be used in our reference implementation.

Chapter 2

Background

This chapter provides the background to the context of this thesis. Section 2.1 gives a brief introduction to the foundations of Formal Concept Analysis, Section 2.2 describes the terminology, Section 2.3 describes important privacy concepts, and Section 2.4 introduces the area of privacy enhancing identity management systems.

2.1 Formal concept analysis

In human thinking and reasoning, *concepts* play an important role. They provide the unit for logic in communication. Thoughts are addressable, since each concept labels a group of objects with specific attributes. A single concept may subsume other concepts and may be subsumed by others.

Formal Concept Analysis (FCA) has been applied, for instance, in the field of knowledge representation, exploration, and software engineering, before, as well as for linguistics. In this section, we quote Ganter and Wille's definitions from [GW99] which are relevant for this thesis and give own examples.

As pointed out in [GW99], *Formal Concept Analysis* (FCA) bases on the mathematization of concept and *conceptual hierarchy*. This algebraic approach provides a formal way

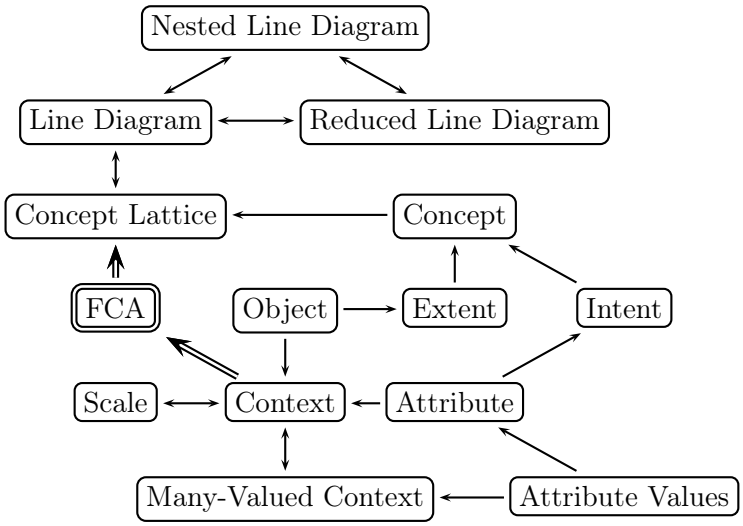


Figure 2.1: FCA Outline

to discover concepts within their hierarchy from a given data source. A concept consists of an *extent* and the corresponding *intent*. Extents consist of *objects*, while intents consist of *attributes*.

We use these names when speaking of this algebraic structure, since they are also used in literature. The background of these names is not important for this thesis. In theory every *thing* might have certain attributes and is therefore a reasonable object. Nevertheless, it makes sense to choose object and attribute sets carefully depending on the analysis objective.

The relation between objects and corresponding attributes is called *incidence relation* and can be described by a cross-table, called *context*. It is the most common starting point for Formal Concept Analysis in order to retrieve formal concepts. The rows of a formal context describe objects and columns attributes, correspondingly.

The informal overview in Figure 2.1 shows the connections between different terms which are used in FCA. A *context*, for instance, is the common input, whereas the *concept lattice*, then, is the common result of FCA. This lattice consists of *concepts*, each consisting of an *extent*, a set of *objects*, and an *intent*, a set of *attributes*. This is defined in a formal

manner in Definition 1.

Definition 1 A *formal context* $\mathbb{K} := (G, M, I)$ consists of two sets G and M and a relation I between G and M. The elements of G are called the *objects* and the elements of M are called the *attributes* of the context. In order to express that an object g is in a relation I with an attribute m, we write gIm or $(g, m) \in I$ and read "the object g *has* the attribute m". [GW99] □

Within a cross-table we can refer to rows or columns. In order to formalize a row or a column, we use the *derivation* operator, a' and b', upon objects $a \in G$ and attributes $b \in M$. Common entries of several rows or columns can be referred by the derivation of a set of objects or attributes.

Definition 2 For a set $A \subseteq G$ of objects we define

$$A' := \{m \in M \mid gIm \text{ for all } g \in A\}$$

(the set of attributes common to the objects in A). Correspondingly, for a set B of attributes we define

$$B' := \{g \in G \mid gIm \text{ for all } m \in B\}$$

(the set of objects which have all attributes in B). [GW99] □

Then, a formal concept provides a set of objects A and a set of attributes B where A have to be equivalent to the derivation of B and vice versa. Thus, A is the set of these objects which have all attributes in B in common. And B is the set of these attributes which are common to all objects in A.

Definition 3 A *formal concept* of the context (G, M, I) is a pair (A, B) with $A \subseteq G$, $A' = B$ and $B' = A$. We call A the *extent* and B the *intent* of the concept (A, B). $\mathfrak{B}(G, M, I)$ denotes the set of all concepts of the context (G, M, I). [GW99] □

Concepts are comparable, i. e. they have an order. This order can be reduced to the set order of either the intent or the extent. This is valid, since a concepts extent and intent are not independent from each other. In fact, the greater the extent set is, the smaller is the intent set and vice versa.

Definition 4 If (A_1, B_1) and (A_2, B_2) are concepts of a context, (A_1, B_1) is called a *subconcept* of (A_2, B_2), provided that $A_1 \subseteq A_2$ (which is equivalent to $B_2 \subseteq B_1$). In this case, (A_2, B_2) is a *superconcept* of (A_1, B_1), and we write $(A_1, B_1) \leq (A_2, B_2)$. The relation $\leq$ is called the *hierarchical order* (or simply *order*) of the concepts. The set of all concepts of (G, M, I) ordered in this way is denoted by $\underline{\mathfrak{B}}(G, M, I)$ and is called the *concept lattice* of the context (G, M, I). [GW99] □

Lattices can be illustrated by line diagrams. A *line diagram* is an undirected graph where each concept is represented by a small circle and a unique label. Labels are the concept's object and attribute sets. Subconcepts are connected by ascending edges to their superconcept and vice versa. The lattice can be retrieved from a corresponding line diagram, since all concepts are included with their objects and attributes as well as the order through the subconcept-superconcept-relation by ascending and descending edges.

Usually, *reduced line diagrams* are used to visualize lattices. The difference to ordinary line diagrams is the labeling of concepts. Labels are formed from objects and attributes whereas each object and each attribute appears just once within the entire line diagram. This can be achieved by denoting each attribute at the concept with most objects which, nevertheless, still includes the attribute. Correspondingly, each object is denoted at the concept with most attributes which still includes the object. Also in case of an ordinary line diagram, it is possible to retrieve the complete concept lattice from a reduced line diagram.[GW99] Thus, we do not distinguish between reduced and ordinary line diagrams and further call both *line diagram*.

Example 1 We observe a channel with a single bit which is assumed to be superposition coding of a decimal number. We denote the states of the bit with set (2^0) and unset

dec	$\overline{2^0}$	2^0
0	×	
1		×

Table 2.1: Binary superposition (1 Bit)

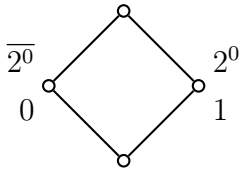


Figure 2.2: Binary superposition (1 Bit, line diagram)

($\overline{2^0}$). Then, the relation between bit state and decimal number is given by Table 2.1. The corresponding lattice is shown in Figure 2.2 and consists of the following concepts:

$$\begin{aligned}\mathfrak{B} = \Big\{ &\big(\{0,1\}, \varnothing\big),\\ &\big(\{0\}, \{\overline{2^0}\}\big),\\ &\big(\{1\}, \{2^0\}\big),\\ &\big(\varnothing, \{\overline{2^0}, 2^0\}\big)\Big\}\end{aligned}$$

In Figure 2.2, the topmost circle represents the first concept which describes attributes shared by all objects. The bottommost circle represents the last concept which describes objects sharing all attributes. □

In reduced line diagrams, the intent of a concept can be retrieved by following all ascending edges starting from the concept circle. Attributes denoted at reachable concept circles belong to the intent of the concept. Correspondingly, the extend of a concept can be retrieved by following descending edges.

Many-valued attributes and scales

Instead of cross-table based contexts[1], we can make use of *many-valued* contexts. Attribute values are, therefore, extended to several values besides crosses.

Definition 5 A *many-values context* (G, M, W, I) consists of sets G, M, and W and a ternary relation I between G, M, and W (i. e., $I \subseteq G \times M \times W$) for which it holds that

$$(g, m, w) \in I \text{ and } (g, m, v) \in I \text{ always implies } w = v$$

The elements of G are called *objects*, those of M *(many-valued) attributes* and those of W *attribute values*.

$(g, m, w) \in I$ we read as "the attribute m has the value w for the object g". (G, M, W, I) is called a *n-valued context*, if W has n elements. The many-valued attributes can be regarded as partial maps from G in W. Therefore, it seems reasonable to write $m(g) = w$ instead of $(g, m, w) \in I$. The *domain* of an attribute m is defined to be

$$\mathrm{dom}(m) := \left\{ g \in G \;\middle|\; (g, m, w) \in I \text{ for some } w \in W \right\}$$

The attribute m is called *complete*, if $\mathrm{dom}(m) = G$. A many-valued context is *complete*, if all its attributes are complete. [GW99] □

Scales are used to reduce the complexity to one-valued contexts and compute the concept lattice. A scale has the same structure like a context, i. e. it has objects and attributes. The object set of a scale has to be equivalent to the set of attribute values of the corresponding many-valued attribute in the many-valued context.

Definition 6 A *scale* for the attribute m of a many-valued context is a (one-valued) context $\mathbb{S}_m := (G_m, M_m, I_m)$ with $m(G) \subseteq G_m$. The objects of a scale are called *scale values*, the attributes are called *scale attributes*. [GW99] □

[1] one-valued contexts [GW99]

The process of *plain scaling* is done by replacing the many-valued attribute by the scales attribute set. Each value can then be represented by the corresponding row from the scale. There are also other scaling methods, such as *relational scaling*, for instance, which is described in [PW99].

To motivate the usage of many-valued contexts, we enhance the previous cross-table and provide the corresponding line diagram. Then, we show how many-valued contexts and plain scaling can be used to arrange concepts more concisely in line diagrams.

Example 2 We reuse Example 1, but this time, we observe two bits with states $\overline{2^1}$, $\overline{2^0}$, 2^0, and 2^1. Thus, four decimals can be coded, confer Table 2.2. We provide a corresponding line diagram in Figure 2.3. Assume, our observation was not complete, i. e. we know only one of two bits. It is then easy to find all numbers, which can be considered as sent, by starting from the concept circle labeled with the observed bit and following all descending edges.

The count of concepts raises from four to ten. If we consider more than two bits, it would more and more become hard to keep track of the presented issue. Therefore, we introduce a (theory driven) scaling in Table 2.3. This scaling of one bit reduces the count of attributes to two, but demands a many-valued context at the same time.

By neglecting the attributes values in the many-valued context, we achieve a line diagram like in Figure 2.2 on page 9. The same applies to the line diagram of the scale in Table 2.3. Thus, we can draw a nested line diagram, confer Figure 2.4, where the outer diagram represents the scale and the inner diagram represents the remaining semantic of the many-valued context. Filled concept circles of inner diagrams are connected to corresponding circles of reachable (inner) diagrams by following edges of the outer diagram. Unfilled circles do not represent a concept of the corresponding lattice. Again, we can deal with an incomplete observation, since considering only one bit can be done by scaling this bit and neglect the inner diagrams. The result would be a diagram like Figure 2.2 which is labeled with two objects at each, the leftmost and rightmost concept circle. □

dec	$\overline{2^1}$	$\overline{2^0}$	2^0	2^1
0	×	×		
1	×		×	
2		×		×
3			×	×

Table 2.2: Binary superposition (2 Bit)

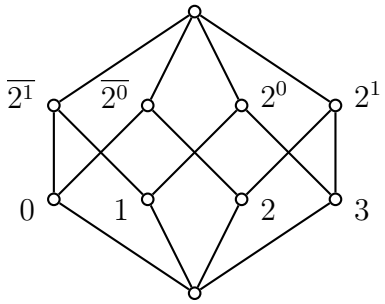


Figure 2.3: Binary superposition (2 Bit, line diagram)

dec	$\overline{2^0}$	2^0
0	$\overline{2^1}$	
1		$\overline{2^1}$
2	2^1	
3		2^1

	$\overline{2^1}$	2^1
$\overline{2^1}$	×	
2^1		×

Table 2.3: Binary superposition (2 Bit, scaled) and scale

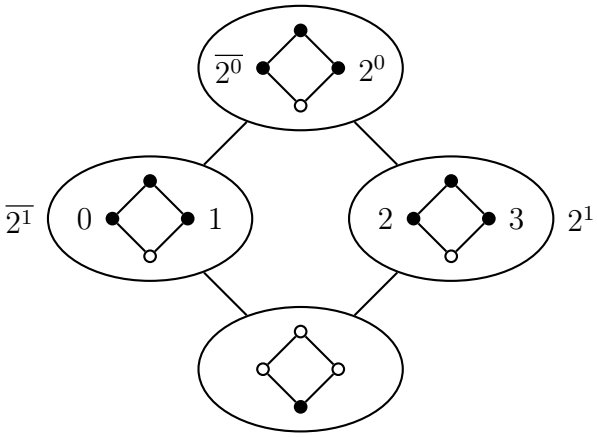


Figure 2.4: Binary superposition (2 Bit, scaled line diagram)

dec	$\overline{2^0}$	2^0
0	$\overline{2^1}\overline{2^2}$	
1		$\overline{2^1}\overline{2^2}$
2	$2^1\overline{2^2}$	
3		$2^1\overline{2^2}$
4	$\overline{2^1}2^2$	
5		$\overline{2^1}2^2$
6	$2^1 2^2$	
7		$2^1 2^2$

	$\overline{2^1}$	2^1
$\overline{2^1}\overline{2^2}$	$\overline{2^2}$	
$2^1\overline{2^2}$		$\overline{2^2}$
$\overline{2^1}2^2$	2^2	
$2^1 2^2$		2^2

	$\overline{2^2}$	2^2
$\overline{2^2}$	×	
2^2		×

Table 2.4: Binary superposition (3 Bit, scaled) and scales

Example 3 We reasoned in Example 2 that neglecting inner diagrams of nested line diagrams would lead to a concise diagram. However, in the resulting diagram, attributes provided by one specific bit have not been taken in account.

For this example, we consider three bits, thus, six states $\overline{2^i}, 2^i$ as attributes with $i \in \{1, 2, 3\}$. Scaling can be done by using similar scales like in Example 2. However, unlike in Example 2, one scale shall be many-valued, too, confer Table 2.4. However, using nested line diagrams within nested line diagrams would fail in a graphical manner.

An opportunity to provide the full range of attributes and conciseness is to zoom a line diagram. A scale or a nested line diagram is used to present a certain subset of attributes and provide a concise view. The characterization of a specific object is, then, done by using further scales, confer Figure 2.5. □

2.2 Terminology

Anonymity, linkability, and observability

Definition 7 *Anonymity* is the state of being not identifiable within a set of subjects, the *anonymity set.* [PH06] □

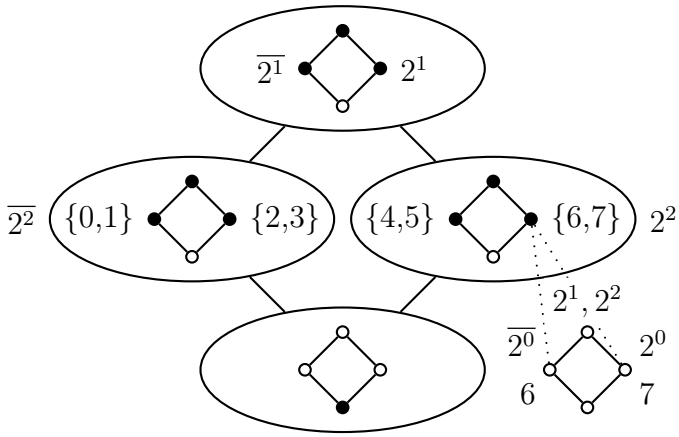


Figure 2.5: Binary superposition (3 Bit, zoomed line diagram)

According to [PH06], a subject which is not identifiable within an anonymity set cannot be *characterized* unambiguously within this set. The anonymity set consists of all subjects which are suspected for having common attributes. Obviously, the size of an anonymity set depends on the observed action and also on background knowledge about subjects.

Characterizations are based on *attributes*. Thus, a unique characterization have to be based on attributes which are unique within the given set. If a subject is identifiable within a set we speak of an *identifiability set* [PH06].

Example 4 Assume an attacker who chooses a graphical representation of subjects to observe them. He is able to distinguish subjects by their shapes which change depending on the subject's attributes.

If the observed subject set contains subjects which have all the same shape, then the attacker is not able to distinguish them at all, sketched in Figure 2.6.

In contrast, subjects which appear with a unique shape are unambiguously characterizable within this set, thus, identifiable by attributes they have. In Figure 2.7 one subject can be distinguished, since it appears not as circle like all the others do. □

The anonymity of a subject depends on these attributes which are of interest for an attacker. In networks with messages between subjects the sender may be of interest as well

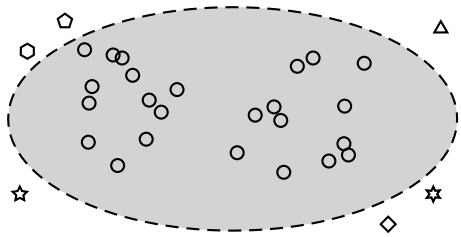


Figure 2.6: Example of an anonymity set

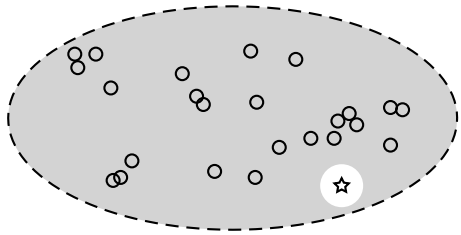


Figure 2.7: Example of an identifiability set

as the recipient. More precisely, the link between message and sender resp. recipient may be of interest. Thus, if an attacker is interested in the interplay of "items of interest"[PH06], these items have to be unlinkable with the subject to keep it anonymous.

Definition 8 (Unlinkability) Unlinkability of two or more items of interest (e. g., subjects, messages, events, actions, ...) means that within the system (comprising these and possibly other items), from the attacker's perspective, these items of interest are no more and no less related after his observation than they are related concerning his a-priori knowledge.[PH06] □

Definition 9 (Linkability) Linkability is the negation of unlinkability, i. e., items are either more or are either less related than they are related concerning the a-priori knowledge. [PH06] □

Example 5 More than one subject send messages anonymously to a network. An attacker observes that two of the messages are of the same size. Obviously, two items of interest have been successfully linked. However, neither the messages nor their size are linkable to

any subject by this knowledge. In particular, messages with the same size do not imply generally same senders or recipients. □

In addition, subjects may use *pseudonyms*. The trivial cases are either to use exactly one pseudonym instead of an identifier of the subject (ID) or to use a previously unused pseudonym for each transaction. The first case provides only anonymity, if the pseudonym is not linkable to the subject. In the latter case anonymity is provided for each transaction where the used pseudonym is not linkable to the subject. Additionally, if the used pseudonym is linkable to another item of interest, for instance another pseudonym, then this item must not be linkable to the subject.

Apart from these two cases, the choice of a specific pseudonym can be taken arbitrarily complex. It may depend on the choice of communication partner as well as on internal subject states, such as the role or the context the user is working in at this very moment.

Pseudonyms are not necessarily bound to a single subject. They can be shared within groups of subjects. Such pseudonyms may be linkable to several or even to all group members. However, actions still remain anonymous, since all group members are, then, in the same anonymity set of suspected originators.

Definition 10 (Pseudonymous) Being pseudonymous is the state of using a pseudonym as ID.[PH06] □

So far, we assumed that items of interest are observable. However, preventing an attacker from observing these items would also lead to anonymity.

Definition 11 (Unobservability) Unobservability is the state of items of interest (IOI) being indistinguishable from any IOI (of the same type) at all.[PH06] □

2.3 Related work on anonymity and linkability

Díaz, Seys, Claessens, and Preneel suggested in [DSCP02] a model which allows to rate schemes for anonymous connections by the degree of anonymity which they provide. This

approach is basing on information theory, whereas the degree of anonymity is computed using probabilities which describe the likelihood that a specific subject sent a particular message. An advantage of this approach is that attacker knowledge about the user can be taken into account, in addition to the amount of users acting within the system. This amount has been the main parameter in previous work about anonymity measures. However, for a data subject it is hard to use these probabilities, since they are based on attacker knowledge which can, usually, just be guessed by another person. In addition, this model is useful to compare different schemes with respect to the degree of anonymity, but a high degree of anonymity within an entire system does not imply necessarily a sufficient degree of anonymity for a data subject.

Steinbrecher and Köpsell proposed in [SK03] a probabilistic approach of formalizing unlinkability. They use, therein, probabilities to describe the relation between two items, where an item can be, for instance, a message or a subject. This relation can be extended to sets of items. The probabilities are used to define a degree of unlinkability in general and a degree of anonymity, in case one item is an identifier. In this paper, the authors also pointed out that *contents of messages* within a communication system can reduce anonymity for a user, in the worst case to zero.

Fischer-Hübner describes in [FH01] an approach to compute the risk of re-identification. The attack model describes an attacker who tries to identify subjects by using a database containing personal data. Her assumption is that identity data has been vanished from personal data within the database and, therefore, all obvious links to identities are lost. An entropy is defined for attributes, where each attribute can have several attributes. Using this entropy, the *average number of values* which are useful for the purpose of re-identification, the *average number of value combinations* are defined successively, and, finally, the *risk of re-identification* basing on the latter one. This approach takes into account that disclosures of different data items not necessarily have the same effect on the anonymity of a data subject. However, the entire database about personal data from

subjects within the system is required to compute the entropy which is required to compute the risk of re-identification. A preventive computation by a data subject within a huge communication system might, therefore, be very hard.

In contrast to Díaz, Seys, Claessens, and Preneel, we do not consider the degree of anonymity which can be provided by a system. Our approach focuses instead on anonymity which is provided for particular users within a communication system. Within the range of attributes which determine the likelihood of linkability between messages and subjects, we focus on similarity between message contents, and thus, similarity between sets of data items. This similarity can then be used as a component to compute probabilities in order to support the approach of Steinbrecher and Köpsell. Fischer-Hübner, however, draw a limit of our approach. In the worst case, two messages, might be linkable by this identity which can be derived from their message contents, whereas these contents are completely different from each other. In such a case, message contents lead both to the re-identification of one and the same subject, i. e. the risk of re-identification is too high for both messages, whereas the similarity between their contents is zero. However, the risk of re-identification alone cannot be used to draw all conclusions about linkability, except it is the risk of re-identification of only one subject. This is not the case in [FH01].

2.4 Privacy enhancing identity management

An *identity management* system keeps track of all personal data items which occur within the communication of a person with communication partners. Sets of these data items which became known to a particular communication partner and also linkable, due to the use of the same pseudonym for instance, are called *partial identities*. A *privacy-enhancing* identity management supports a person in choosing pseudonyms and data items such that privacy goals like data-minimization and unlinkability can be achieved best while the communication still serves the purpose. In particular, the step from revealing a *partial* identity

to revealing the complete *civil* identity is a threat for privacy, in case the user performs this step unintentionally.

PRIME project

In the European Union, laws limit the risk of data exploration, but a trustful enforcement is still missing for the digital communication, as it has been pointed out in [HK05]. The PRIME project addresses this gap. A general overview provides [FHAH05]. One objective of this project is to build a privacy enhancing identity management system in order to support persons to keep their privacy against other parties, if so intended. Another objective is to build an identity management system which provides a legal, trustworthy way of managing personal data of customers within companies or public corporations in order to develop the foundation for a system which is multilaterally secure.

The PRIME project has yet not finished its work.

Chapter 3

FCA applied to communication

In this chapter, we show how Formal Concept Analysis can be applied to communication, in particular, to the privacy related parts of communication.

First, we introduce how deducible data can be represented, define then the representation of messages as a container for data items, and refine, then, the description of messages with attributes of connection control data and the connection context.

3.1 Data lattices

In this section we introduce the representation of *data items* and the correlations between them.

As a first approach, we explore two data items, *drivers licence* and *sufficient driving experiences*. Given, a person with a drivers licence has been proved for driving skills in an exam and has, therefore, sufficient driving experiences. Then, nobody drives without a licence and, thus, everybody with sufficient driving experiences has a drivers licence.

We are going to represent this relation in a formal context in order to compute a concept lattice using FCA. We use *data items* as objects *and* attributes. The attributes of a formal concept will describe these *data items* which are deducible from data items

	licence	experiences
licence	×	×
experiences	×	×

Table 3.1: Equivalence of two data items

in the corresponding object set. Both, sufficient driving experiences and having a drivers licence, are equivalent in our limited view, since the licence implies experiences and vice versa. A cross-table which represents this setting is, therefore, completely filled by crosses, confer Table 3.1. In this basic case, we obtain one formal concept, the only concept in the corresponding concept lattice, which represents both *data items*. It represents the equivalence of both *data items*, since they are both contained in the object set as well as in the attribute set:

$$\mathfrak{B} = \big\{(\{\text{licence}, \text{experiences}\}, \{\text{licence}, \text{experiences}\})\big\}$$

We take a look on a more complex example. Given, the *drivers licence* provides not only the fact of having a licence, but include the complete licence paper. A European drivers licence, for instance, includes *name* and *date and place of birth*. We want to compare it with another official document and consider the German identity card. It includes *name*, *date and place of birth*, *nationality*, *address*, *height*, *color of eyes*, and *religious name or pseudonym*. We present this *contained-in* relation between data items as a cross-table in Table 3.2. Twelve formal concepts can be obtained when computing the concept lattice for this cross-table:

	d	i	n	b	y	a	h	c	p
d	×		×	×					
i		×	×	×	×	×	×	×	×
n			×						
b				×					
y					×				
a						×			
h							×		
c								×	
p									×

d – drivers licence
i – German identity card
n – name
b – date and place of birth
y – nationality
a – address
h – height
c – color of eyes
p – religious name or pseudonym

Table 3.2: Drivers licence and German identity card

$$\begin{aligned}\mathfrak{B} = \Big\{ &\big(\{d,i,n,b,y,a,h,c,p\},\varnothing\big),\\ &\big(\{d,i,n\},\{n\}\big),\\ &\big(\{d,i,b\},\{b\}\big),\\ &\big(\{i,y\},\{y\}\big),\\ &\big(\{i,a\},\{a\}\big),\\ &\big(\{i,h\},\{h\}\big),\\ &\big(\{i,c\},\{c\}\big),\\ &\big(\{i,p\},\{p\}\big),\\ &\big(\{d,i\},\{n,b\}\big),\\ &\big(\{d\},\{d,n,b\}\big),\\ &\big(\{i\},\{i,n,b,y,a,h,c,p\}\big),\\ &\big(\varnothing,\{d,i,n,b,y,a,h,c,p\}\big)\Big\}\end{aligned}$$

We use line diagrams to represent concept lattices and present in Figure 3.1 a corresponding diagram for the actual lattice.

From this lattice, we can obtain, for instance, which data items are sufficient for deanonymization, if the *name* and the *address* are sufficient for identification. These are exactly

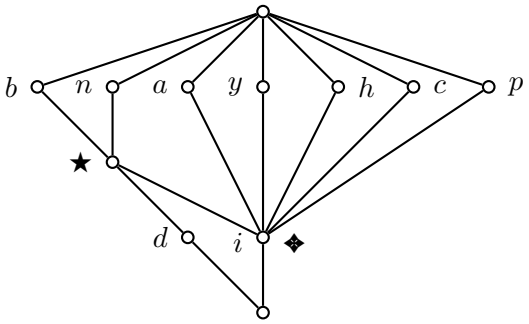


Figure 3.1: Data lattice for drivers licence and German identity card

	d	i	n	b	y	a	h	c	p
d	×		×	×					
i		×	×	×	×	×	×	×	×
v			×			×			
n			×						
b				×					
y					×				
a						×			
h							×		
c								×	
p									×

d – drivers licence
i – German identity card
v – business card
n – name
b – date and place of birth
y – nationality
a – address
h – height
c – color of eyes
p – rel. name or pseudonym

Table 3.3: Extension by *business card*

the object items found in the greatest concept which contains n and a within its attribute set. In our case, ❖ in Figure 3.1, $(\{i\}, \{i, n, b, y, a, h, c, p\})$, is the greatest concept with corresponding attributes. Accordingly, only the *identity card* can be used to reveal the person's identity.

In case that it is necessary to reveal *name* and *address* at the same time, but nothing else, we need to consider another object in the cross-table. This new object is either used as a dummy or provides an own data item, in addition to deducible data items.

To provide such a new object in the current example, we extend the previous example by a data item and consider this item as *business card*. It is meant to provide only *name* and *address*, but no own data items, confer Table 3.3. The corresponding line diagram in Figure 3.2 shows the location of the new concept in a similar lattice as Figure 3.1. The

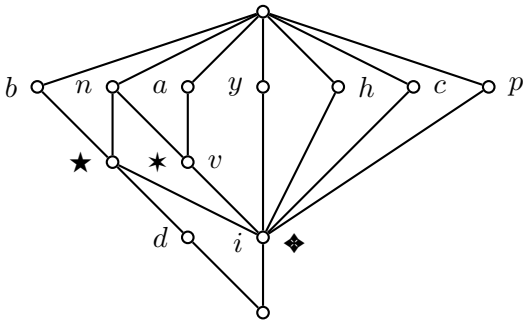


Figure 3.2: Data lattice for extension by *business card*

concept lattice consists of thirteen concepts,

$$\begin{aligned}\mathfrak{B} = \Big\{ &\big(\{d,i,v,n,b,y,a,h,c,p\},\varnothing\big),\\ &\big(\{d,i,v,n\},\{n\}\big),\\ &\big(\{d,i,b\},\{b\}\big),\\ &\big(\{i,y\},\{y\}\big),\\ &\big(\{i,v,a\},\{a\}\big),\\ &\big(\{i,h\},\{h\}\big),\\ &\big(\{i,c\},\{c\}\big),\\ &\big(\{i,p\},\{p\}\big),\\ &\big(\{d,i\},\{n,b\}\big),\\ &\big(\{i,v\},\{n,a\}\big),\\ &\big(\{d\},\{d,n,b\}\big),\\ &\big(\{i\},\{i,n,b,y,a,h,c,p\}\big),\\ &\big(\varnothing,\{d,i,n,b,y,a,h,c,p\}\big)\Big\}\end{aligned}$$

The new concept $(\{i,v\},\{n,a\})$, ✶ in Figure 3.2, is greater than ❖ and, therefore, the greatest concept which contains n and a in its attribute set. Thus, it is now sufficient for

deanonymization to get either the *identity card* or the *business card*.

The appropriate concept for this request is easy to find in the representation of this concept lattice as line diagram. Descending lines are traced starting from the two circles which are labeled by n and a. The highest circle in the diagram in which the lines cross represents the greatest concept which contains both attributes.

In another request, we assume *name* and *date and place of birth* to be sufficient for identification. Then, *identity card* or the *drivers licence* are sufficient for deanonymization. This fact is provided by the greatest concept which contains n and b is $(\{d, i\}, \{n, b\})$, $\bigstar$ in Figure 3.1 and 3.2.

3.1.1 Data types

We can add dummies to the *attribute* set of the context as well as we added the *business card* dummy to the object set in the previous example. We call such an attribute *data type*. It is used to summarize a set of data items.

For instance, a person may have two or more addresses. Each address would, then, be represented by a different data item, even if the provided information is quite similar to other addresses. To stress the fact that they provide the same *type* of information, we can use a data type *address* which summarizes all addresses in the actual context. As well as in natural language, we obtain a superconcept to all addresses which is labeled by the data type *address*.

To reuse the previous example, we assume two addresses instead of one which are summarized by the data type *address*, confer Table 3.4. We alter the related data items for *identity card* and *business card* to provide the different addresses. The *primary address* is used for the *identity card* and *secondary address* for *business card*. Both cards have to support the data type, since they include an address, then.

In this updated example, we can still find the statements of the previous examples. For instance, the *name* and the *date and place of birth* can be either provided by the *drivers*

	d	i	n	b	y	a_1	a_2	h	c	p	A
d	×		×	×							
i		×	×	×	×	×		×	×	×	×
v			×				×				×
n			×								
b				×							
y					×						
a_1						×					×
a_2							×				×
h								×			
c									×		
p										×	

d – drivers licence
i – German identity card
v – business card
n – name
b – date and place of birth
y – nationality
a_1 – primary address
a_2 – secondary address
h – height
c – color of eyes
p – rel. name or pseudonym
A – address data type

Table 3.4: Data type *address*

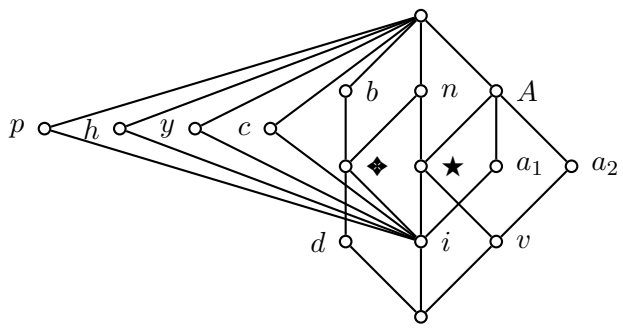


Figure 3.3: Data lattice for data type *address*

licence or by the *identity card*. This information is provided by the concept $(\{d, i\}, \{n, b\})$, ❖ in Figure 3.3. The same way provides $(\{i, v\}, \{n, A\})$, ★ in Figure 3.3, that the *name* together with any *address* can be revealed using either *identity card* or the *business card*.

Additionally, we can, now, distinguish between *primary* and *secondary* address, if necessary. So, there is $(\{i\}, \{i, n, b, y, a_1, h, c, p, A\})$ in the lattice, labeled by i in Figure 3.3, as well as $(\{v\}, \{n, a_2, A\})$, labeled by v. Both are the smallest concepts which contain the *name* and one of the addresses in their attribute set. Thus, *name* and *primary address* is deducible only from the *identity card*, whereas *name* and *secondary address* is deducible only from the *business card* in this example.

	d	v	n	a
m_1			×	
m_2				×
m_3	×			
m_4		×		

(a) one-valued

	data item
m_1	n
m_2	a
m_3	d
m_4	v

(b) many-valued

	d	n	b	a
m_1		×		
m_2				×
m_3	×	×	×	
m_4		×		×

(c) scaled using Table 3.3

$m_1, \ldots, m_4$ – messages
d – drivers licence
v – business card
n – name
b – date and place of birth
a – address

Table 3.5: Messages and data items

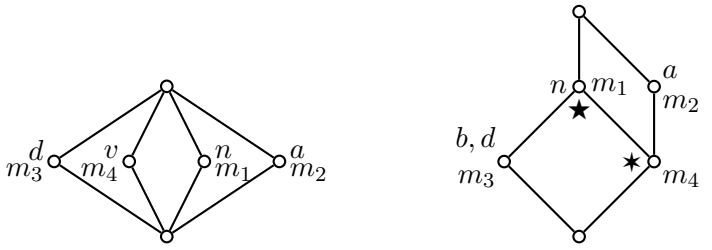
3.2 Messages

So far, we showed how dependencies and connections between data items can be represented in formal context and concept lattices. We assumed that data items are sent in one run and no further communication takes place between communicating people. In this section, we extend the model and consider *messages* in order to cover also communication relations that last longer than one data exchange. Initially, we assume that messages are not correlated or linkable with each other.

The main purpose of a message is to *contain* data items. It can be used to sum up over data items of the whole or a part of a communication relation where several messages have been sent. In addition, we use messages to assign other attributes, such as an *originator* and a *recipient* in following sections.

We choose *messages* as formal objects and *data items* as formal attributes to achieve an appropriate definition of formal contexts. The incidence relation represents, then, which data items are contained in a message.

As an example, we choose a person who revealed the *name* in one message, the *address* in a second one, in the third message the *drivers licence*, and in a forth the *business card*. Then, the formal context includes four objects, each message is one, and four attributes, confer Table 3.5(a).



(a) plain (b) scaled using Table 3.3

Figure 3.4: Lattice for messages and data items

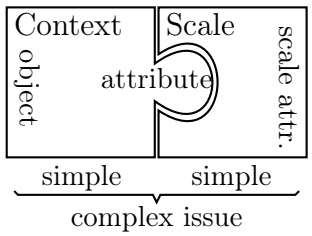


Figure 3.5: Outline "plain scaling"

The concept lattice represents a non-overlapping separation of data items with respect to messages, confer Figure 3.4(a). Each data item has been sent just once in the example, thus, there must yet not be any overlap. However, deducible data items can be redundant between the messages.

In order to include deducible data items, we can use the context from Table 3.3 on page 24 as a formal scale to obtain all deducible data items from the many-valued context in Table 3.5(b).

Then, only the first and second message contain still single data items. In the third message, the *address* and *date and place of birth* have to be added to the *drivers licence*. In the fourth message, we have to replace *business card* by *name* and *address* which are deducible according to Table 3.3. Then, the scaled context and the corresponding lattice show a more realistic relation with data items shared by different messages, confer Table 3.5(c) and Figure 3.4(b).

Correlations between message contents can be obtained from the concept lattice. For

instance, the first, third, and fourth message contain the *name* as data item. The corresponding concept is $(\{m_1, m_3, m_4\}, \{n\})$, ★ in Figure 3.4(b). However, *name* and *address* together are only included in the fourth message, since the greatest concept including both data items as attributes is $(\{m_4\}, \{n, a\})$ resp. ✶.

We can also ask for all messages which are correlated by the same data items as a given set of messages. Such a message set can be $\{m_3, m_4\}$, for instance. Then, we look for the smallest concept which includes m_3 and m_4 as objects. In our example, it is exactly $(\{m_1, m_3, m_4\}, \{n\})$, ★ in Figure 3.4(b). The common data item is n which is provided by m_1, in addition to the other two messages. Thus, all three messages are found to be *correlated* by an attacker who discovers the contents correlation between m_3 and m_4. Then, each correlation between messages is an asset with respect to *linkability of messages* for an attacker who is able to compare their contents, confer Definition 8 and 9 on page 15.

3.3 Connection control data

In the previous section, we proposed attributes for messages besides contents. In particular, *connection control data* is interesting to look for further correlations between messages. It covers data about the state of the connection which was used to send a message. For instance, it can contain a flag which makes private messages distinguishable from public readable messages. We discuss *supplementary flags* like this after first introducing the representation of *subjects* in general and *originator* and *recipient* in particular.

3.3.1 Subjects and messages

A subject acts either as originator or recipient of a message. In a following section, we introduce even more subject roles. For this section, we assume that the difference between such subject roles is not important, i. e. all subjects belong to the same role.

A subject is related to the message, if it *knows* a message. We can add this relation

	d	n	b	a	$\mathbb{S}_1$	$\mathbb{S}_2$
m_1		×				×
m_2				×	×	
m_3	×	×	×		×	
m_4		×		×		×

(a) one-valued

	d	n	b	a	subject
m_1		×			$\mathbb{S}_2$
m_2				×	$\mathbb{S}_1$
m_3	×	×	×		$\mathbb{S}_1$
m_4		×		×	$\mathbb{S}_2$

(b) many-valued

$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{S}_1, \mathbb{S}_2$ – subjects

Table 3.6: Message context enhanced by subjects

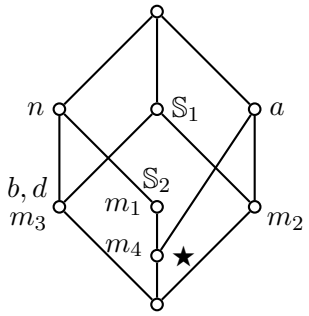
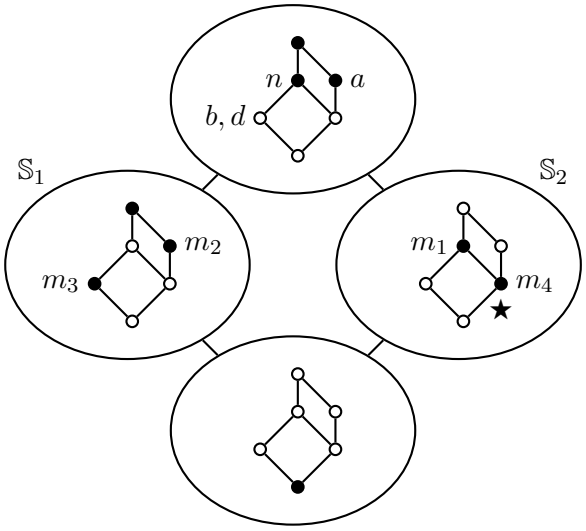


Figure 3.6: Message lattice enhanced by subjects

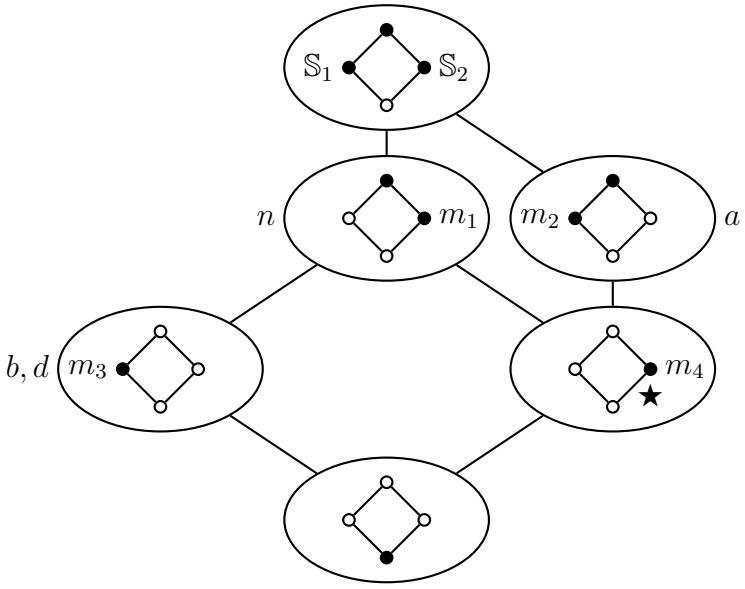
to the formal context of the previous section. We get a context with *messages* as formal objects and *data items* as well as *subjects* as formal attributes. We can speak of the *contained in* relation with respect to *messages* and *data items* and of the *known by* relation with respect to *messages* and *subjects*.

We enhance the context from Table 3.5(c) with two subjects, where the second subject knows the first and fourth message and the first subject knows the second and third message, confer Table 3.6. Different line diagrams can be found in Figure 3.6 and 3.7. All of them show the same lattice which corresponds to Table 3.6(a).

The lattice concepts still provide the correlations in message contents. This can be seen in Figure 3.7 which reuse the structure of Figure 3.4(b) on page 29. In addition, these correlations are now differentiated with respect to the subjects. For instance, $(\{m_4\}, \{n, a, \mathbb{S}_2\})$,



(a) Attributes in the outer diagram are subjects



(b) Attribute in the outer diagram are data items

Figure 3.7: Nested message lattices (equivalent to Figure 3.6)

★ in Figure 3.6 and 3.7, is the greatest concept which includes n and a in its attribute set. Thus, only the fourth message contains *name* and *address*. Furthermore, we know that the fourth message is known by the second subject, since $\mathbb{S}_2$ is included in the concept's attribute set. Since $\mathbb{S}_1$ is not included, it follows that the greatest concept including n, a, and $\mathbb{S}_1$ is $(\varnothing, \{d, n, b, a, \mathbb{S}_1, \mathbb{S}_2\})$. Thus, we know, there is no message which contains *name* and *address* and is known by the first subject.

However, it is not possible to conclude from a single concept of this lattice that only the second subject knows *name* and *address*. The formal context which we define in this section does not offer any relation between the different attribute types like *data items* and *subjects*. Thus, there is no concept which states that a *subject* knows a specific *data item*. However, we show how to develop a lattice which deals with the complete knowledge of subjects after introducing pseudonyms in the next section.

3.3.2 Pseudonyms and subjects

So far, we assumed subjects to be equivalent to identities in reality. With respect to a technical background like the Internet, the usage of pseudonyms is more realistic, instead.

To introduce pseudonyms, we can replace the subjects by pseudonyms in Section 3.3.1. The only difference is the interpretation. Thus, all relations between messages and *subjects* which can be obtained as consequence of Section 3.3.1 are also valid as relations between messages and *pseudonyms*. We can reconstruct the relation between messages and subjects, if we know the relation between pseudonyms and subjects. For this reconstruction, we have to scale the formal context which contains pseudonyms with respect to the relation between pseudonyms and subjects.

We reuse the many-valued context of Table 3.6(b) for the current example, but consider pseudonyms instead of subjects, confer Table 3.7. In Table 3.8, we provide two example relations between pseudonyms and subjects. By scaling with the context in Table 3.8(a), for instance, we obtain the example context which is already known from the previous

	d	n	b	a	pseudonym
m_1		×			$\mathbb{P}_4$
m_2				×	$\mathbb{P}_1$
m_3	×	×	×		$\mathbb{P}_3$
m_4		×		×	$\mathbb{P}_2$

$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms

Table 3.7: Message context and pseudonyms

	$\mathbb{S}_1$	$\mathbb{S}_2$
$\mathbb{P}_1$	×	
$\mathbb{P}_2$		×
$\mathbb{P}_3$	×	
$\mathbb{P}_4$		×

(a)

	$\mathbb{S}_1$	$\mathbb{S}_2$	$\mathbb{S}_3$	$\mathbb{S}_4$
$\mathbb{P}_1$	×			×
$\mathbb{P}_2$		×		
$\mathbb{P}_3$	×		×	
$\mathbb{P}_4$	×		×	×

(b)

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
$\mathbb{S}_1, \ldots, \mathbb{S}_4$ – subjects

Table 3.8: Pseudonym–Subject scales

section.

Table 3.8(b) provides a more complex scale. In general, arbitrary relations between pseudonyms and subjects are possible. In our example, we present four cases. The subject uses only one pseudonym, the second subject in this case. The subject use several pseudonyms, the first subject in, for instance. Several subjects share one pseudonym, the first, third, and fourth subject, for instance. And several subjects share several pseudonyms, the first and third subject, for instance. We present a corresponding scale lattice in Figure 3.8.

The scale also shows how the pseudonyms are correlated. The fourth pseudonym is the one which is used by most subjects. Three subjects use it, one of them is the first subject

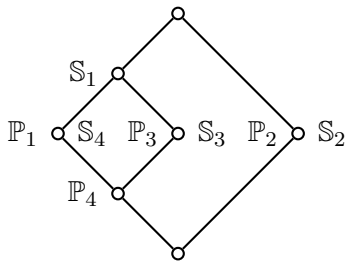


Figure 3.8: Lattice for Pseudonym–Subject scale in Table 3.8(b)

	d	n	b	a	$\mathbb{S}_1$	$\mathbb{S}_2$	$\mathbb{S}_3$	$\mathbb{S}_4$
m_1		×			×		×	×
m_2				×	×			×
m_3	×	×	×		×		×	
m_4		×		×		×		

$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{S}_1, \ldots, \mathbb{S}_4$ – subjects

Table 3.9: Message context with applied pseudonym scale

which controls most of the pseudonyms, i. e. the first, third, and fourth.

We provide the scaled message context in Table 3.9 and a nested line diagram of the corresponding lattice in Figure 3.9. This lattice can be used to find correlations between messages like the lattices in Section 3.3.1.

3.3.3 Subjects and data items

Unfortunately, no relations between subjects and data items can be obtained from the previously introduced lattices, so far. In Section 3.3.1, we have seen that the lattice in Figure 3.6 respective 3.7 was not sufficient to draw a conclusion about connections between subjects and data items, since both are formal attributes and none of them an object in the corresponding context.

For this section, we choose *subjects* as formal objects and *data items* as attributes. This relation cannot be found in examples of previous sections. However, we introduced already contexts with relations between *messages* and *subjects* in Section 3.3.1 and 3.3.2. From these contexts, we obtain the *subject–data items* relation, by inverting the context with respect to objects and attributes and scaling *messages* using the relation between *messages* and *data items*.

We need to satisfy two requirements for *plain scaling*, confer Definition 6 and the following paragraph. First, the attribute which we want to scale has to be *many-valued* in the given context. And second, all *attribute values* of this many-valued attribute have to occur in the scale as objects. From the contexts in Section 3.3.1 and 3.3.2, we derive the relations in Table 3.10(a) and 3.10(c). The messages are still multiple one-valued attributes

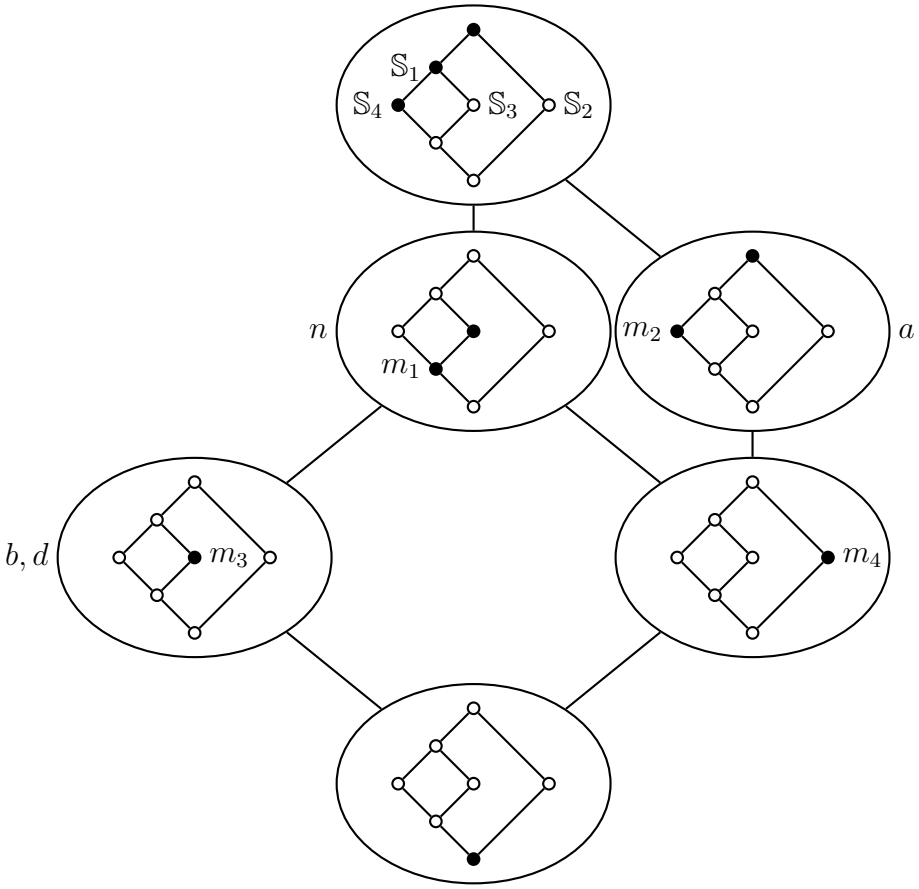


Figure 3.9: Lattice for applied pseudonym scale

	m_1	m_2	m_3	m_4
$\mathbb{S}_1$		×	×	
$\mathbb{S}_2$	×			×

(a) one-valued

	messages
$\mathbb{S}_1$	$\{m_2, m_3\}$
$\mathbb{S}_2$	$\{m_1, m_4\}$

(b) many-valued

	m_1	m_2	m_3	m_4
$\mathbb{S}_1$	×	×	×	
$\mathbb{S}_2$				×
$\mathbb{S}_3$	×		×	
$\mathbb{S}_4$	×	×		

(c) one-valued

	messages
$\mathbb{S}_1$	$\{m_1, m_2, m_3\}$
$\mathbb{S}_2$	$\{m_4\}$
$\mathbb{S}_3$	$\{m_1, m_3\}$
$\mathbb{S}_4$	$\{m_1, m_2\}$

(d) many-valued

$m_1, \ldots, m_4$ – messages
$\mathbb{S}_1, \ldots, \mathbb{S}_4$ – subjects

Table 3.10: Subject–Message relation in Table 3.6 and 3.9

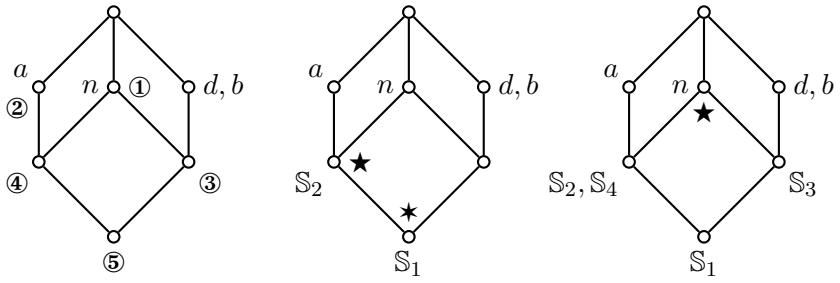
	d	n	b	a	label
$\{m_1\}$		×			①
$\{m_2\}$				×	②
$\{m_3\}$	×	×	×		③
$\{m_4\}$		×		×	④
$\{m_1, m_2\}$		×		×	④
$\{m_1, m_3\}$	×	×	×		③
$\{m_1, m_4\}$		×		×	④
$\{m_2, m_3\}$	×	×	×	×	⑤
$\{m_2, m_4\}$		×		×	④
$\{m_3, m_4\}$	×	×	×	×	⑤
$\{m_1, m_2, m_3\}$	×	×	×	×	⑤
$\{m_1, m_2, m_4\}$		×		×	④
$\{m_1, m_3, m_4\}$	×	×	×	×	⑤
$\{m_2, m_3, m_4\}$	×	×	×	×	⑤
$\{m_1, m_2, m_3, m_4\}$	×	×	×	×	⑤

$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address

Table 3.11: Message scale

in these contexts. We summarize them by replacing the different message attributes by one many-valued attribute where values are chosen from the power set of $\{m_1, m_2, m_3, m_4\}$, i. e. of the set of all possible message sets. We present the corresponding many-valued contexts in Table 3.10(b) and 3.10(d).

The second requirement for *plain scaling* determines the way we have to define the objects of the *scale*. Actually, it is only required that the scale objects include all attribute values of the corresponding attribute in the context. We choose a more sophisticated way to use one fixed scale for both example contexts. This is possible, since the relation between *messages* and *data items*, which is going to be represented by the scale, bases on Table 3.5(c) on page 28 and is the same in both examples. To be sure to cover all possible attribute values, we use the power set of all messages as scale objects. The scale attributes are, then, all *data items* which are related to any message in the object message set. We present the scale in Table 3.11 and the scale lattice in Figure 3.10(a). The context lattices in Figure 3.10(b) and 3.10(c) have the same structure like the scale lattice, since there are no additional attributes in these contexts which are not covered by this specific scale.



(a) scale (b) Table 3.10(b) (c) Table 3.10(d)

Figure 3.10: Lattice of message scale and scaled contexts

We can now look for relations between subjects and data items. In Section 3.3.1, for instance, it was not possible to conclude from the concept lattice that the first subject knows also *name* and *address*. We can find a message which contains the address and another one containing the name, both known by the first subject. The objective was to express this knowledge from both messages in one concept. Now, in Figure 3.10(b) on page 38, we find $(\{\mathbb{S}_1, \mathbb{S}_2\}, \{n, a\})$, labeled by ★, which shows that both subjects know both, *name* and *address*. Two more results can be obtained from the concept lattice. First, the second subject does not know more than name and address, since ★ is the smallest concept which includes $\mathbb{S}_2$. And second, the first subject knows every single data item we considered in the context, since $(\{\mathbb{S}_1\}, \{d, n, b, a\})$, labeled by ✶, is the smallest concept in the entire lattice.

In the example of Section 3.3.2 it was harder to read the correlation between subjects from Figure 3.9. Now, with the lattice in Figure 3.10(c), we can also draw simple conclusions about the subject correlation for this example. For instance, the second and fourth subject know only *name* and *address*. Thus, both know the same data, even though the fourth subject knows more messages than the second. It is also obvious that each of the four subjects knows the name, since $(\{\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \mathbb{S}_4\}, \{n\})$, ★ in Figure 3.10(c), is the greatest concept which contains n.

These context definitions can be used to compute concept lattices which reveal correla-

	$\mathbb{O}$	$\mathbb{R}$
$\mathbb{S}_1$		×
$\mathbb{S}_2$		×
$\mathbb{S}_3$		×
$\mathbb{S}_4$	×	

$\mathbb{S}_1, \ldots, \mathbb{S}_4$ – subjects
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role

Table 3.12: Subject–Role assignment

	d	n	b	a	$\mathbb{O}$	$\mathbb{R}$
$\mathbb{S}_1$	×	×	×	×		×
$\mathbb{S}_2$		×		×		×
$\mathbb{S}_3$	×	×	×			×
$\mathbb{S}_4$		×		×	×	

$\mathbb{S}_1, \ldots, \mathbb{S}_4$ – subjects
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role

Table 3.13: United context with subject roles

tions or links, respectively, between *subjects* respective *pseudonyms*, instead of messages.

3.3.4 Originator and recipient

In the previous sections, subjects (or pseudonyms) have not been differentiated in *originators* and *recipients*. This is useful as long as we can assume all subjects to act in the same *subject role*, i. e. either *originator* or *recipient*. We show in this section how a context can be constructed which include several subject roles at once.

Fixed subject roles

We speak of *fixed* subjects roles, if we can assume that *subject roles* are independent from messages and, therefore, only depend on the specific subject or pseudonym which is described by the *subject role*. Such an assignment is given in Table 3.12. It can be used to extend the previous example contexts by subject roles.

This extension is done by uniting a context with the recently introduced assignment context. For example, we choose the context in Table 3.10(d), scale it with Table 3.11, and unite it with the subject–role assignment in Table 3.12. The result is Table 3.13.

It is now clear that the second and the fourth subject know *name* and *address*, since

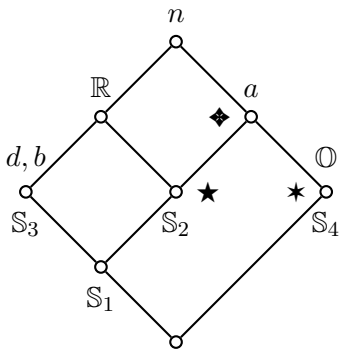


Figure 3.11: Lattice for united context with subject roles

	d	n	b	a	$\mathbb{O}$	$\mathbb{R}$
m_1		×			$\mathbb{P}_1$	$\mathbb{P}_3$
m_2				×	$\mathbb{P}_2$	$\mathbb{P}_1$
m_3	×	×	×		$\mathbb{P}_1$	$\mathbb{P}_4$
m_4		×		×	$\mathbb{P}_3$	$\mathbb{P}_1$

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role

Table 3.14: Message context enhanced by originator and recipient

we can find $(\{\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_4\}, \{n, a\})$ in the lattice, ❖ in Figure 3.11. However, in addition, we conclude from $(\{\mathbb{S}_1, \mathbb{S}_2\}, \{n, a, \mathbb{R}\})$, ★ in Figure 3.11, that the second subject knows the data items from a received message, whereas $(\{\mathbb{S}_4\}, \{n, a, \mathbb{O}\})$, ✶ in Figure 3.11, indicates that the fourth subject knows the data items, since it sent them in a message.

Context definitions which are defined in this section refine the description of subjects. We can compute the lattice from such a context which can be used to find finer grained correlations between subjects.

Subject role depending on message

In a more general approach, we consider subject roles which *vary* with respect to subjects *and* messages. Like in Section 3.3.2, we extend the scaled message context in Table 3.5(c) on page 28 by pseudonyms, but this time, we use two many-valued attributes instead of one, one for the *originator* and the other one for the *recipient*, confer Table 3.14.

So far, it is not possible to scale even just one of the many-valued attributes to one-

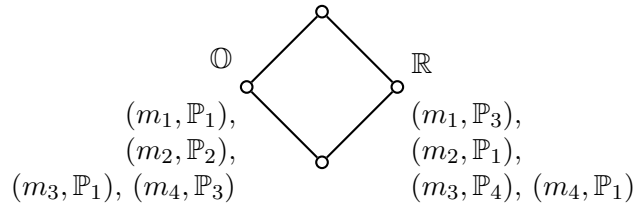


Figure 3.12: Lattice for plain relational context

valued attributes, since neither *originator* nor *recipient* depend only on the formal object, but on a message *and* a subject. Therefore, we choose tuples as formal objects, each consisting of a message and a subject. Then, it is possible to express subject roles as one-valued attributes, confer Table 3.15(a).

Without further attributes, we achieve a partitioning of the previously defined tuples with respect to the subject roles, confer Figure 3.12. All tuples which are related to the same subject role belong, then, to the same concept in this lattice.

More correlations between these tuples can be explored by considering more attributes. For example, we can consider the pseudonym from the object tuple as attribute, confer Table 3.15(b). Then, we can conclude from the concept lattice which messages have been *sent* or *received* by a specific pseudonym, confer the lattice in Figure 3.13. The first pseudonym knows four messages, for instance, since

$$(\{(m_1, \mathbb{P}_1), (m_2, \mathbb{P}_1), (m_3, \mathbb{P}_1), (m_4, \mathbb{P}_1)\}, \{\mathbb{P}_1\})$$

is the greatest concept which contains $\mathbb{P}_1$ as attribute. The pseudonym has been used as originator for two of these messages and the other two messages have been received, since the greatest concepts containing $\mathbb{P}_1$ and either $\mathbb{O}$ or $\mathbb{R}$ as attribute are

$$\big(\{(m_1, \mathbb{P}_1), (m_3, \mathbb{P}_1)\}, \{\mathbb{P}_1, \mathbb{O}\}\big)$$
$$\big(\{(m_2, \mathbb{P}_1), (m_4, \mathbb{P}_1)\}, \{\mathbb{P}_1, \mathbb{R}\}\big)$$

	$\mathbb{O}$	$\mathbb{R}$
$(m_1, \mathbb{P}_1)$	×	
$(m_1, \mathbb{P}_3)$		×
$(m_2, \mathbb{P}_2)$	×	
$(m_2, \mathbb{P}_1)$		×
$(m_3, \mathbb{P}_1)$	×	
$(m_3, \mathbb{P}_4)$		×
$(m_4, \mathbb{P}_3)$	×	
$(m_4, \mathbb{P}_1)$		×

(a) plain

	$\mathbb{P}_1$	$\mathbb{P}_2$	$\mathbb{P}_3$	$\mathbb{P}_4$	$\mathbb{O}$	$\mathbb{R}$
$(m_1, \mathbb{P}_1)$	×				×	
$(m_1, \mathbb{P}_3)$			×			×
$(m_2, \mathbb{P}_2)$		×			×	
$(m_2, \mathbb{P}_1)$	×					×
$(m_3, \mathbb{P}_1)$	×				×	
$(m_3, \mathbb{P}_4)$				×		×
$(m_4, \mathbb{P}_3)$			×		×	
$(m_4, \mathbb{P}_1)$	×					×

(b) pseudonyms

	d	n	b	a	$\mathbb{O}$	$\mathbb{R}$
$(m_1, \mathbb{P}_1)$		×			×	
$(m_1, \mathbb{P}_3)$		×				×
$(m_2, \mathbb{P}_2)$				×	×	
$(m_2, \mathbb{P}_1)$				×		×
$(m_3, \mathbb{P}_1)$	×	×	×		×	
$(m_3, \mathbb{P}_4)$	×	×	×			×
$(m_4, \mathbb{P}_3)$		×		×	×	
$(m_4, \mathbb{P}_1)$		×		×		×

(c) data items

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role

	d	n	b	a	$\mathbb{P}_1$	$\mathbb{P}_2$	$\mathbb{P}_3$	$\mathbb{P}_4$	$\mathbb{O}$	$\mathbb{R}$
$(m_1, \mathbb{P}_1)$		×			×				×	
$(m_1, \mathbb{P}_3)$		×					×			×
$(m_2, \mathbb{P}_2)$				×		×			×	
$(m_2, \mathbb{P}_1)$				×	×					×
$(m_3, \mathbb{P}_1)$	×	×	×		×				×	
$(m_3, \mathbb{P}_4)$	×	×	×					×		×
$(m_4, \mathbb{P}_3)$		×		×			×		×	
$(m_4, \mathbb{P}_1)$		×		×	×					×

(d) pseudonyms and data items

Table 3.15: Relational context with originator and recipient

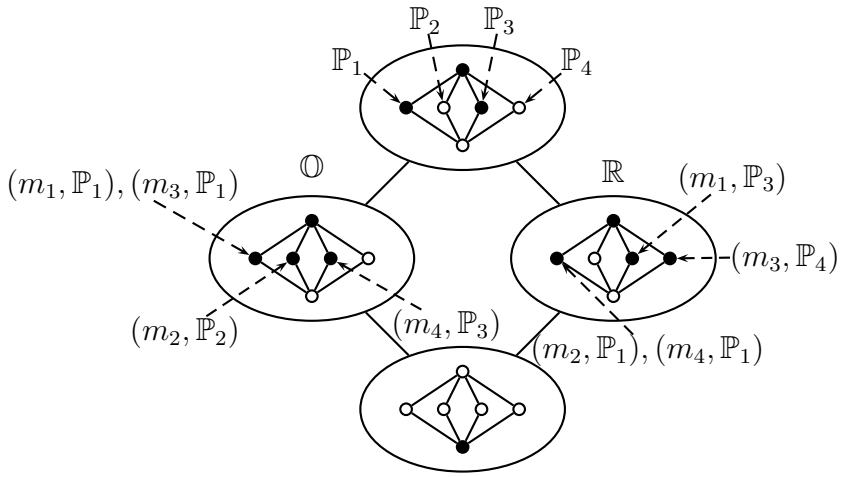


Figure 3.13: Lattice for relational context with pseudonyms

The fourth pseudonym, however, has been used for receiving only, since the greatest concept which contains $\mathbb{P}_4$ as attribute is

$$(\{(m_3, \mathbb{P}_4)\}, \{\mathbb{P}_4, \mathbb{R}\})$$

and contains also $\mathbb{R}$.

In addition, it is possible to consider the message in each tuple as an attribute. However, the correlations are, then, quiet obvious, since there are exactly two object tuples for each message in the context, one for the originator, the other one for the recipient. Nevertheless, we can scale the attribute messages with Table 3.5(c) on page 28, as we did previously in Section 3.3.3. The result is a context with data items and subject roles as attributes, confer Table 3.15(c). The conclusions which can be drawn from a corresponding lattice, like Figure 3.14, are quite similar to conclusions in Section 3.2. Each concept represents data items which are common in different messages. In addition, we get the pseudonyms from the concept's object set which correspond to the messages like in Section 3.3.1 and we differentiate them by subject roles.

For instance, the first and the third pseudonym have been used to sent messages contain-

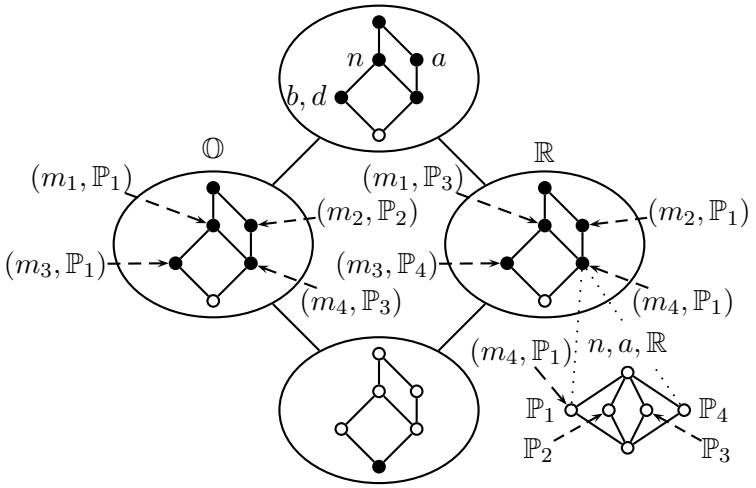


Figure 3.14: Lattice for relational context with data items

ing the *name*. The greatest concept containing n and $\mathbb{O}$ is $(\{(m_1, \mathbb{P}_1), (m_3, \mathbb{P}_1), (m_4, \mathbb{P}_3)\}, \{n, \mathbb{O}\})$. So, we can conclude at the same time which messages have been involved.

If we consider pseudonyms as attributes, in addition, the concept lattice represents which pseudonyms share specific sets of data items as we did in Section 3.3.1. The enhanced context is given in Table 3.15(d) and the lattice extension is drawn in the zoomed part of Figure 3.14.

These contexts which we define in this section refine the description of messages, in particular the description of related subjects. They can be used to compute lattices which are useful to find correlations between messages.

Pseudonyms, data items, and subject roles

Similar to the situation before introducing the context definitions in Section 3.3.3 we are not able to summarize all data items in one concept which have been *sent* or *received* by a specific subject.

We use *relational scaling* as described in [PW99] for the context in Table 3.14 on page 40 which allows to omit messages in object tuples. It defines a combination of *subject* and

	m_1	m_2	m_3	m_4	$\mathbb{O}$	$\mathbb{R}$
$(\mathbb{P}_1, \mathbb{O})$	×		×		×	
$(\mathbb{P}_1, \mathbb{R})$		×		×		×
$(\mathbb{P}_2, \mathbb{O})$	×				×	
$(\mathbb{P}_2, \mathbb{R})$						×
$(\mathbb{P}_3, \mathbb{O})$				×	×	
$(\mathbb{P}_3, \mathbb{R})$	×					×
$(\mathbb{P}_4, \mathbb{O})$					×	
$(\mathbb{P}_4, \mathbb{R})$			×			×

(a) one-valued

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
$m_1, \ldots, m_4$ – messages
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role

	messages	$\mathbb{O}$	$\mathbb{R}$
$(\mathbb{P}_1, \mathbb{O})$	$\{m_1, m_3\}$	×	
$(\mathbb{P}_1, \mathbb{R})$	$\{m_2, m_4\}$		×
$(\mathbb{P}_2, \mathbb{O})$	$\{m_1\}$	×	
$(\mathbb{P}_2, \mathbb{R})$			×
$(\mathbb{P}_3, \mathbb{O})$	$\{m_4\}$	×	
$(\mathbb{P}_3, \mathbb{R})$	$\{m_1\}$		×
$(\mathbb{P}_4, \mathbb{O})$		×	
$(\mathbb{P}_4, \mathbb{R})$	$\{m_3\}$		×

(b) many-valued

Table 3.16: Subject–message relation with subject roles

subject role as tuple, and *subject roles* and *messages* as formal attributes, confer Table 3.16.

The following procedure is similar to that in Section 3.3.3. We have to scale this context in order to replace the message sets by data items. We can use Table 3.11 on page 37 as scale, since we make use of the same relation between *messages* and *data items* like in Section 3.3.3. The lattice which, then, contains the relation between pseudonyms and data items is defined by the scaled context, confer Table 3.17. A corresponding line diagram can be found in Figure 3.15. The object labels in this line diagram do not contain the second tuple component, i. e. the subject role, since this is already covered by the corresponding attributes.

In this lattice, we do not find any relation to messages, anymore. However, we can draw conclusions about correlations between pseudonyms with respect to data items which have

	d	n	b	a	$\mathbb{O}$	$\mathbb{R}$
$(\mathbb{P}_1, \mathbb{O})$	×	×	×		×	
$(\mathbb{P}_1, \mathbb{R})$		×		×		×
$(\mathbb{P}_2, \mathbb{O})$		×			×	
$(\mathbb{P}_2, \mathbb{R})$						×
$(\mathbb{P}_3, \mathbb{O})$		×		×	×	
$(\mathbb{P}_3, \mathbb{R})$		×				×
$(\mathbb{P}_4, \mathbb{O})$					×	
$(\mathbb{P}_4, \mathbb{R})$	×	×	×			×

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role

Table 3.17: Scaled subject–message relation with subject roles

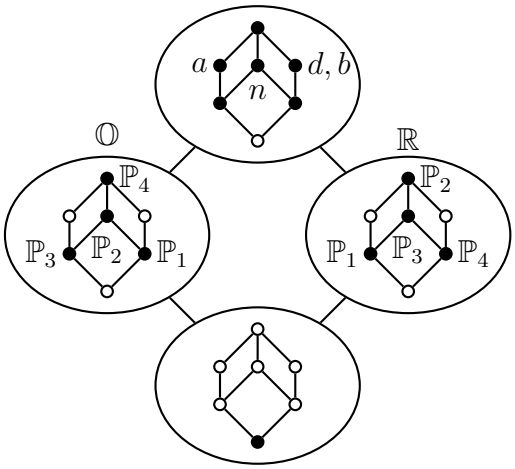


Figure 3.15: Lattice for data items and subjects with subject roles

been either *sent* or *received*. For instance, we find $(\{(\mathbb{P}_3, \mathbb{O})\}, \{n, a, \mathbb{O}\})$ as the greatest concept which contains *name* and *address* in its attribute set as *sent* data items. Thus, the third pseudonym is the only one which has been used to send *name* and *address*, whereas the first pseudonym also knows about the data items, since there is a greater concept $(\{(\mathbb{P}_1, \mathbb{R}), (\mathbb{P}_3, \mathbb{O})\}, \{n, a\})$ which contains the data items and does not limit the pseudonyms to originators or recipients.

Correspondingly, we can determine the data items which have been sent or received by a specific pseudonym. The second pseudonym, for instance, has not received any data item, since the smallest concept which contains $(\mathbb{P}_2, \mathbb{R})$ in its object set is

$$(\{(\mathbb{P}_1, \mathbb{R}), (\mathbb{P}_2, \mathbb{R}), (\mathbb{P}_3, \mathbb{R}), (\mathbb{P}_4, \mathbb{R})\}, \{\mathbb{R}\})$$

The fourth pseudonym sent *address*, *drivers licence*, *name*, and *date and place of birth*. Thus, it sent each data item which has been sent by the second pseudonyms and more.

In this section we refined the context definition by adding subject roles as attributes to subjects respective pseudonyms. The lattices computed from these contexts can be used to find finer grained correlations between subjects respective pseudonyms.

3.3.5 Supplementary flags

So far, we discussed the essentials of connection control data. At the beginning of Section 3.3 we proposed an indicator which makes *private* distinguishable from *public* messages by a *supplementary flag*. Private messages should be exclusively known to the originator and receiver, whereas public messages are possibly readable by everyone. We explain this example and give the general procedure of adding supplementary flags at the end of this section.

Obviously, a flag which indicates public messages is an attribute to messages. So, we can add this attribute to context definitions starting from Section 3.2. When introducing

	$\mathbb{S}_1$	$\mathbb{S}_2$	$\mathbb{S}_3$	$\mathbb{S}_4$
$\mathbb{P}_1$	×			×
$\mathbb{P}_2$		×		
$\mathbb{P}_3$	×		×	
$\mathbb{P}_4$	×		×	×
$\mathbb{P}_p$	×	×	×	×

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
$\mathbb{P}_p$ – public message pseudonym
$\mathbb{S}_1, \ldots, \mathbb{S}_4$ – subjects

Table 3.18: Enhanced Pseudonym–Subject scale

subjects in Section 3.3, however, we have to take into account that the content of public messages become known to every subject. After introducing pseudonyms in Section 3.3.2, this can be done by adding another pseudonym instead of a flag attribute to the specific context which is used to scale pseudonyms to subjects. This new pseudonym has, then, to be related to all subjects. The scale in Table 3.8(b) on page 34 can be enhanced as shown in Table 3.18 by adding such an additional pseudonym $\mathbb{P}_p$.

The example context in Section 3.3.3 has to be altered in a different way. All messages which are flagged as public have to become related to all objects in this context, i. e. to all subjects. The scale in Table 3.11 on page 37 can be left untouched provided it already covers all messages, public as well as private ones.

When considering subject roles, such as originator or recipient, we can proceed right the same way. In Table 3.14 on page 40, we have to introduce a new pseudonym which is, then, used for all public messages. The use of this pseudonym makes, in fact, only sense in the recipient role. However, it does only affect the lattice at all, if we scale pseudonyms to subjects like in Section 3.3.2. For this scaling, the new pseudonym has to be related to all subjects.

We can introduce further flags in a similar way. First, we have to identify which context can be enhanced by the chosen flag. Then, we have to apply the effect of a set or unset flag and alter the contexts correspondingly. Finally, the enhanced lattice provides concepts which take the conceptual differentiation by the specific flag into account. Further attributes will be introduced in the next section.

3.4 Communication context

In addition to connection control data, it is possible to consider the circumstances under which a message has been sent. We call these circumstances *communication context* which is subject of this section. Usually, the communication context is determined by preliminary knowledge and preceding negotiations between parties who are going to exchange data items. It covers, for instance, the purpose of sent messages, obligations which are bound to the message content, and the subject or pseudonym which the content is about.

3.4.1 Data subject

So far, we considered data items which have been sent from an originator to a recipient within a message. We did not consider additional subjects when a message content is not related to originator or recipient but to a third party. In fact, there are messages where the content is neither related to originator nor recipient. A corresponding scenario is a company who got customer data and is going to verify it against a data base of known frauds. If this database is offered as a service by another company, then the customer is the third party which is subject of message contents between both companies. We call this third party *data subject*, according to [PC95, Article 2].

We introduced subjects as attributes of messages when discussing the originator or recipient role in Section 3.3.1. A data subject can be added in general the same way. However, we have to take care that only one kind of subjects is contained in such an enhanced context. In Section 3.3.1 we had the choice between originator or recipient. Now, we can choose data subjects, in addition, while the interpretation of the relation varies depending on the choice. Previously, it was *sent by* or *received by*; and *regarding* applies, then, for data subjects. We can completely adopt the results from Section 3.3.1, since there is no structural change of the defined formal context as long as it is not necessary to mix data subjects with originators or recipients.

If message contents do not regard subjects but pseudonyms, we can conclude about subjects by scaling them right the same way as we did in Section 3.3.2. Of course, this requires to know about the relation between pseudonyms and subjects.

We can even adopt Section 3.3.3 to conclude about which data items are known about a specific subject, altogether. The rearranging and scaling procedure is quite the same. The difference is, again, only the interpretation of the relation between subjects and data items. In Section 3.3.3, it was the *known by* relation and now we can speak of the *known of* relation.

Generally, it is also possible to adopt *fixed subject roles* from Section 3.3.4. However, then, we have to assume that there are only subjects which are either originator, recipient, or data subject. The first two subject roles are conceivable in real world, whereas it is harder to reason the latter one. In real world, a company or person represented by a subject which is fixed to the *data subject role* would exist formally only, since it would just be subject of conversation but would not actively take part in any conversation. However, of course, it is possible to limit the view on the world appropriately from a broader context, i. e. the object or attribute set of a corresponding formal context. Then, the data subject role is just an additional subject role in the assignment in Table 3.12 on page 39. This enhanced assignment can be united with the relation between *subjects* and *data items*, similar to Table 3.13 on page 39.

The same way, we can add this third subject role to contexts which cover *subject roles depending on messages*. First, we have to enhance a context like in Table 3.14 on page 40 by a *third* many-valued attribute for data subjects. They can be transformed to one-valued attributes by constructing object tuples from messages and subjects, as we did in Section 3.3.4, confer Table 3.19. We have to consider that data subject and either the originator or the recipient may be the same. Thus, the structure of corresponding lattices changes compared to the lattice in, for instance, Figure 3.14 on page 44. We can use the advantage of separating the inner and outer structure in this figure. The actual

	d	n	b	a	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$
m_1		×			$\mathbb{P}_1$	$\mathbb{P}_3$	$\mathbb{P}_4$
m_2				×	$\mathbb{P}_2$	$\mathbb{P}_1$	$\mathbb{P}_2$
m_3	×	×	×		$\mathbb{P}_1$	$\mathbb{P}_4$	$\mathbb{P}_3$
m_4		×		×	$\mathbb{P}_3$	$\mathbb{P}_1$	$\mathbb{P}_1$

	d	n	b	a	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	
$(m_1, \mathbb{P}_1)$		×			×			1a
$(m_1, \mathbb{P}_3)$		×				×		1b
$(m_1, \mathbb{P}_4)$		×					×	1c
$(m_2, \mathbb{P}_2)$				×	×		×	2a
$(m_2, \mathbb{P}_1)$				×		×		2b
$(m_3, \mathbb{P}_1)$	×	×	×		×			3a
$(m_3, \mathbb{P}_4)$	×	×	×			×		3b
$(m_3, \mathbb{P}_3)$	×	×	×				×	3c
$(m_4, \mathbb{P}_3)$		×		×	×			4a
$(m_4, \mathbb{P}_1)$		×		×		×	×	4b

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role
$\mathbb{C}$ – data subject role

Table 3.19: Subject roles with data subject

change affects only the outer structure, confer Figure 3.16. The two stars in the diagram show where object tuples appear which contain a subject with data subject role and either originator, ★ in Figure 3.16, or recipient role, ✶ in Figure 3.16.

As an application of this lattice, we can look for all data items which are known from a given message. For instance, just the *name* is known from the first message. This is represented by the smallest concept which includes m_1 in its object set and $\mathbb{C}$ in the corresponding attribute set, i. e.

$$(\{(m_1, \mathbb{P}_4), (m_3, \mathbb{P}_3), (m_4, \mathbb{P}_1)\}, \{n, \mathbb{C}\})$$

It is labeled with 1c in Figure 3.16. In addition, we can conclude that the same data items can be derived from the third and the fourth message. We can even decide which pseudonym has been used to send the data items, since they are also included in the object tuple.

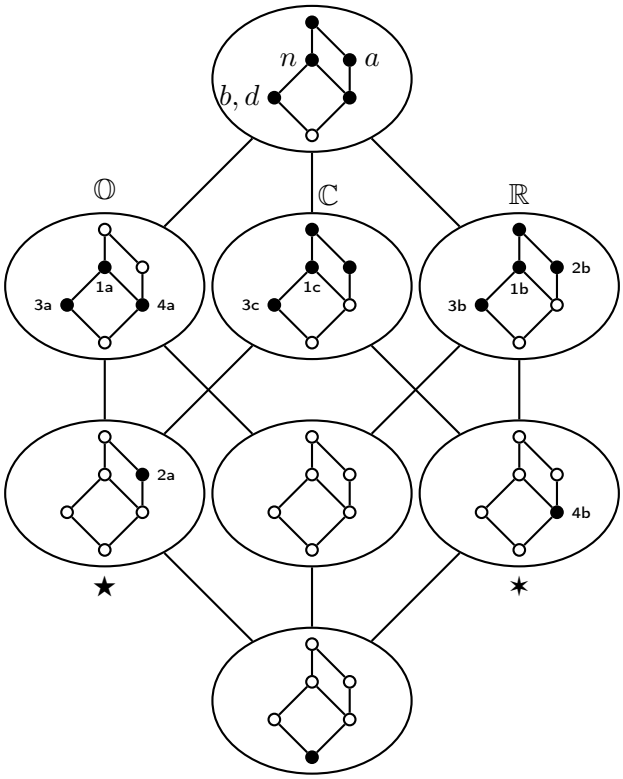


Figure 3.16: Subject role lattice with data subject

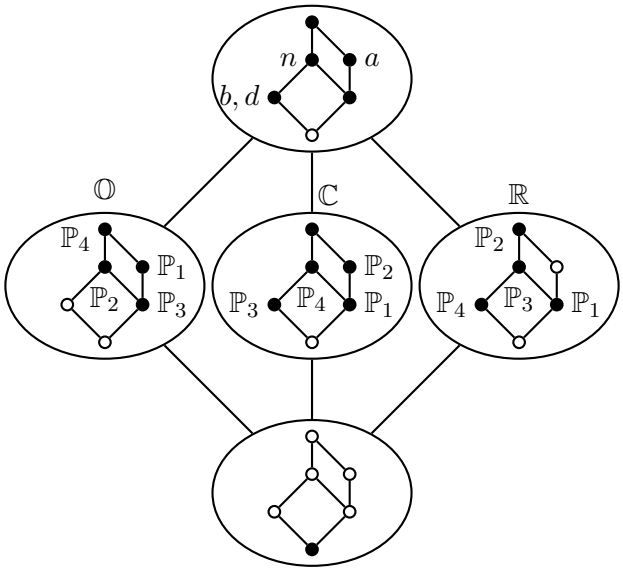


Figure 3.17: Data items and subjects with three subject roles

Like in *Subject role depending on message* of Section 3.3.4, we are lacking in an opportunity to summarize all data items which are known of a specific pseudonym. We use a similar relational scaling like in the paragraph *Pseudonyms, data items, and subjects* to construct a lattice with tuples of subject and subject role as formal objects and data items and subject roles as formal attributes. We use the same context as in the previous paragraph as base, confer Table 3.20. In the line diagram given in Figure 3.17, we separate subject roles and data items, i. e. the different kinds of attribute, in the outer and the inner structure. In this lattice, it is possible to summarize all data items which have been made available independent from messages or which are known from a specific subject in particular. In addition, this lattice reveals which pseudonyms have been in a specific subject role. All pseudonyms have been a *data subject*, for instance. This is represented by the greatest concept which contains $\mathbb{C}$ in its attribute set, $(\{(\mathbb{S}_1, \mathbb{C}), (\mathbb{S}_2, \mathbb{C}), (\mathbb{S}_3, \mathbb{C}), (\mathbb{S}_4, \mathbb{C})\}, \{\mathbb{C}\})$.

We use this example lattice in following sections instead of the lattices presented in Section 3.3.4, since they are subsumed in the actual definition.

	messages	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$
$(\mathbb{P}_1, \mathbb{O})$	$\{m_1, m_3\}$	×		
$(\mathbb{P}_1, \mathbb{R})$	$\{m_2, m_4\}$		×	
$(\mathbb{P}_1, \mathbb{C})$	$\{m_4\}$			×
$(\mathbb{P}_2, \mathbb{O})$	$\{m_1\}$	×		
$(\mathbb{P}_2, \mathbb{R})$			×	
$(\mathbb{P}_2, \mathbb{C})$	$\{m_2\}$			×
$(\mathbb{P}_3, \mathbb{O})$	$\{m_4\}$	×		
$(\mathbb{P}_3, \mathbb{R})$	$\{m_1\}$		×	
$(\mathbb{P}_3, \mathbb{C})$	$\{m_3\}$			×
$(\mathbb{P}_4, \mathbb{O})$		×		
$(\mathbb{P}_4, \mathbb{R})$	$\{m_3\}$		×	
$(\mathbb{P}_4, \mathbb{C})$	$\{m_1\}$			×

(a) many-valued

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role
$\mathbb{C}$ – data subject role

	d	n	b	a	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$
$(\mathbb{P}_1, \mathbb{O})$	×	×	×		×		
$(\mathbb{P}_1, \mathbb{R})$		×		×		×	
$(\mathbb{P}_1, \mathbb{C})$		×		×			×
$(\mathbb{P}_2, \mathbb{O})$		×			×		
$(\mathbb{P}_2, \mathbb{R})$						×	
$(\mathbb{P}_2, \mathbb{C})$				×			×
$(\mathbb{P}_3, \mathbb{O})$		×		×	×		
$(\mathbb{P}_3, \mathbb{R})$		×				×	
$(\mathbb{P}_3, \mathbb{C})$	×	×	×				×
$(\mathbb{P}_4, \mathbb{O})$					×		
$(\mathbb{P}_4, \mathbb{R})$	×	×	×			×	
$(\mathbb{P}_4, \mathbb{C})$		×					×

(b) scaled

Table 3.20: Data items and subjects with three subject roles

3.4.2 Purpose

In Europe, data protection acts consider a *purpose* of data disclosure, if personal data items are processed, confer [PC95, Article 6]. We show how to construct lattices which consider these purposes and, therefore, make them available as an additional attribute which can be used to sort out messages and discover further correlations.

We assume *purposes* to be attributes of messages. Actually, each *message content* is related to a purpose. Therefore, our assumption may fail, if there is more than one content in a message. However, in Section 3.2 we define our messages such that they contain only one content. We can even limit the number of purposes per message to one. Messages which serve several purposes can be split into several messages, one for each purpose. Thus, a message does not need to be related to more than one purpose. Additionally, it is not useful to connect purposes with data items independent from messages, since the same content can be sent multiple times, but with different purposes.

The example context of Section 3.2 can, then, be enhanced by one-valued purpose attributes, one attribute for each purpose. The effect is the same like in Section 3.3.1 when enhancing the message context with subjects. The further enhancement by subjects and pseudonyms as additional attributes can, then, be done like described in Section 3.3.1. This is possible, since the purpose of a message does not depend on a subject.

To enhance subject contexts like defined in Section 3.3.3, however, it is not sufficient just to introduce additional attributes. The purpose has, there, to become part of formal objects to keep it in relation to messages, since messages are, now, attributes according to the definition. We can either add each purpose as additional formal object or construct object tuples, each consisting of a subject and a purpose.

The first opportunity is not very useful. In addition to correlations in the knowledge of subjects, correlations between purposes can, then, be obtained and even between purposes and subjects. We see no practical use in comparing subjects and purposes with respect to data items. The opportunity provided by constructing object tuples, however, leads

	m_1	m_2	m_3	m_4	**a**	**i**	**c**
$(\mathbb{S}_1, \mathbf{a})$	×	×			×		
$(\mathbb{S}_1, \mathbf{i})$			×			×	
$(\mathbb{S}_2, \mathbf{c})$				×			×
$(\mathbb{S}_3, \mathbf{a})$	×				×		
$(\mathbb{S}_3, \mathbf{i})$			×			×	
$(\mathbb{S}_4, \mathbf{a})$	×	×			×		

(a) one-valued

$m_1, \ldots, m_4$ – messages
$\mathbb{S}_1, \ldots, \mathbb{S}_4$ – subjects
d – drivers licence
n – name
b – date and place of birth
a – address
a – acknowledgement
i – verify identity
c – keep in contact

	messages	**a**	**i**	**c**
$(\mathbb{S}_1, \mathbf{a})$	$\{m_1, m_2\}$	×		
$(\mathbb{S}_1, \mathbf{i})$	$\{m_3\}$		×	
$(\mathbb{S}_2, \mathbf{c})$	$\{m_4\}$			×
$(\mathbb{S}_3, \mathbf{a})$	$\{m_1\}$	×		
$(\mathbb{S}_3, \mathbf{i})$	$\{m_3\}$		×	
$(\mathbb{S}_4, \mathbf{a})$	$\{m_1, m_2\}$	×		

(b) many-valued

	d	n	b	a	**a**	**i**	**c**
$(\mathbb{S}_1, \mathbf{a})$		×		×	×		
$(\mathbb{S}_1, \mathbf{i})$	×	×	×			×	
$(\mathbb{S}_2, \mathbf{c})$		×		×			×
$(\mathbb{S}_3, \mathbf{a})$		×			×		
$(\mathbb{S}_3, \mathbf{i})$	×	×	×			×	
$(\mathbb{S}_4, \mathbf{a})$		×		×	×		

(c) scaled

Table 3.21: Subjects and data items w. r. t. the purpose

to useful lattices. We differentiate, then, the knowledge of subjects by purposes. Formal attributes are data items which are known to a certain subject and have been sent with a specific purpose.

As an example, we consider three purposes, *for acknowledgement*, *verify identity*, and *keep in contact*. In Section 3.3.3, the example context from Table 3.10(c) on page 36 contained four messages. We assign the purpose *for acknowledgement* to the first two messages. The third message contains the drivers license and we assume it to be sent for the purpose of *verifying an identity*, whereas the fourth message has been sent including the business card and, we assume, this has been done, for instance, to *keep in contact*.

We enhance Table 3.10(c) on page 36 for our example context in Table 3.21(a). Like in Section 3.3.3, we summarize the message attributes to one many-valued attribute, confer Table 3.21(b) and scale it using Table 3.11 on page 37. The resulting context, presented in Table 3.21(c), is one-valued and can be used to compute the concept lattice. This lattice, we present a corresponding line diagram in Figure 3.18, can be used to draw conclusions

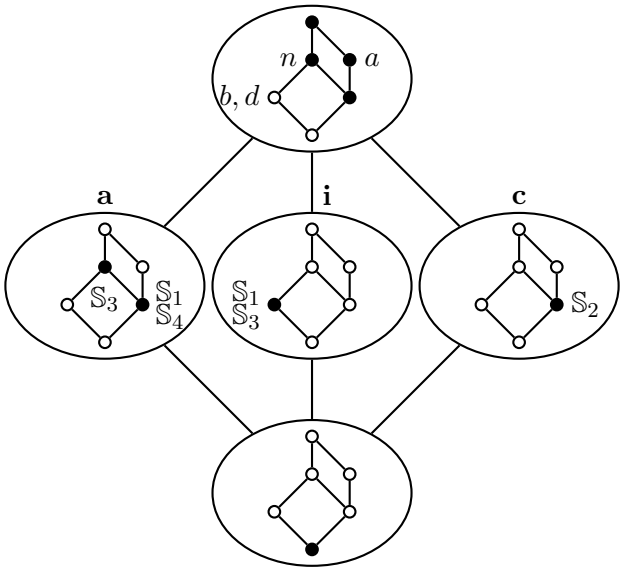


Figure 3.18: Subjects and data items w. r. t. the purpose

about the summarized knowledge of a specific subject differentiated by the purpose which has been assigned to data items.

In Section 3.3.3 we discovered, for instance, that the first, the second, and the fourth subject know about *name* and *address*. The same information is provided by

$$(\{(\mathbb{S}_1, \mathbf{a}), (\mathbb{S}_2, \mathbf{c}), (\mathbb{S}_4, \mathbf{a})\}, \{n, a\})$$

in the current lattice. However, we can now distinguish more precisely between *name* and *address* which have been provided for *acknowledgement* respective for *keeping in contact*. So, there is the concept $(\{(\mathbb{S}_1, \mathbf{a}), (\mathbb{S}_4, \mathbf{a})\}, \{n, a, \mathbf{a}\})$ which summarizes all subjects in its object set which know messages with the purpose *for acknowledgement* about name and address. And there is $(\{(\mathbb{S}_2, \mathbf{c})\}, \{n, a, \mathbf{c}\})$ which summarizes all subjects which know messages with the purpose *keep in contact* about the same data items.

The changes to the first contexts which we introduced in Section 3.3.4 can be adopted

with the additional flags which we mentioned at the beginning of the current section. However, the changes to the context definition from the section about *pseudonyms, data items, and subject roles* are more complex. Again, we are going to scale and transform *messages* to *data items* and, therefore, we have to make the purpose part of the formal object. Thus, the object tuples which have been introduced in Section 3.3.4 have to be enhanced to triples, each consisting of a pseudonym, a subject role, and a purpose.

Instead of the example in Section 3.3.4 we show the actual enhancement using the example of Section 3.4.2, since the first mentioned example is subsumed in the latter one. In addition, we reuse the assignment of purposes of the previous section. The example context can be developed, then, from scaling the message attributes as shown in Table 3.22.

To draw a corresponding line diagram, we use an outer diagram which separates *subject role* attributes and an inner diagram which adds *data item* attributes to concepts, confer Figure 3.19. However, this approach does not necessarily represent the whole lattice, since *purpose* attributes are not considered. They can be added by zooming into a small circle and presenting the missing attributes, as outlined in Figure 3.19.

The zooming process can be done in an arbitrary order, thus, we can exchange outer and inner line diagram or present the zoomed line diagram as outer or inner diagram, instead. For instance, a line diagram which presents *purpose* attributes in its outer diagram is shown in Figure 3.20. The zooming step has, then, to present the missing subject role attributes.

Lattice of purposes

So far, purposes have been assumed to be independent from each other. As pointed out in [FH01], this is not necessarily the case. For instance, in a hospital, personal data can be inquired for *medical treatment*, whereas this covers *treatment of infections* and *cancer treatment*. Thus, purposes provide their own lattice. We can arrange the currently

	msgs	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	**a**	**i**	**c**
$(\mathbb{P}_1, \mathbb{O}, \mathbf{a})$	$\{m_1\}$	×			×		
$(\mathbb{P}_1, \mathbb{O}, \mathbf{i})$	$\{m_3\}$	×				×	
$(\mathbb{P}_1, \mathbb{R}, \mathbf{a})$	$\{m_2\}$		×		×		
$(\mathbb{P}_1, \mathbb{R}, \mathbf{c})$	$\{m_4\}$		×				×
$(\mathbb{P}_1, \mathbb{C}, \mathbf{c})$	$\{m_4\}$			×			×
$(\mathbb{P}_2, \mathbb{O}, \mathbf{a})$	$\{m_1\}$	×			×		
$(\mathbb{P}_2, \mathbb{C}, \mathbf{a})$	$\{m_2\}$			×	×		
$(\mathbb{P}_3, \mathbb{O}, \mathbf{c})$	$\{m_4\}$	×					×
$(\mathbb{P}_3, \mathbb{R}, \mathbf{a})$	$\{m_1\}$		×		×		
$(\mathbb{P}_3, \mathbb{C}, \mathbf{i})$	$\{m_3\}$			×		×	
$(\mathbb{P}_4, \mathbb{R}, \mathbf{i})$	$\{m_3\}$		×			×	
$(\mathbb{P}_4, \mathbb{C}, \mathbf{a})$	$\{m_1\}$			×	×		

(a) many-valued

$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – pseudonyms
$m_1, \ldots, m_4$ – messages
d – drivers licence
n – name
b – date and place of birth
a – address
$\mathbb{O}$ – originator role
$\mathbb{R}$ – recipient role
$\mathbb{C}$ – data subject role
a – acknowledgement
i – verify identity
c – keep in contact

	d	n	b	a	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	**a**	**i**	**c**	label
$(\mathbb{P}_1, \mathbb{O}, \mathbf{a})$		×			×			×			1oa
$(\mathbb{P}_1, \mathbb{O}, \mathbf{i})$	×	×	×		×				×		1oi
$(\mathbb{P}_1, \mathbb{R}, \mathbf{a})$				×		×		×			1ra
$(\mathbb{P}_1, \mathbb{R}, \mathbf{c})$		×		×		×				×	1rc
$(\mathbb{P}_1, \mathbb{C}, \mathbf{c})$		×		×			×			×	1cc
$(\mathbb{P}_2, \mathbb{O}, \mathbf{a})$		×			×			×			2oa
$(\mathbb{P}_2, \mathbb{C}, \mathbf{a})$				×			×	×			2ca
$(\mathbb{P}_3, \mathbb{O}, \mathbf{c})$		×		×	×					×	3oc
$(\mathbb{P}_3, \mathbb{R}, \mathbf{a})$		×				×		×			3ra
$(\mathbb{P}_3, \mathbb{C}, \mathbf{i})$	×	×	×				×		×		3ci
$(\mathbb{P}_4, \mathbb{R}, \mathbf{i})$	×	×	×			×			×		4ri
$(\mathbb{P}_4, \mathbb{C}, \mathbf{a})$		×					×	×			4ca

(b) scaled

Table 3.22: Knowledge w. r. t. pseudonyms, subject role, and purpose

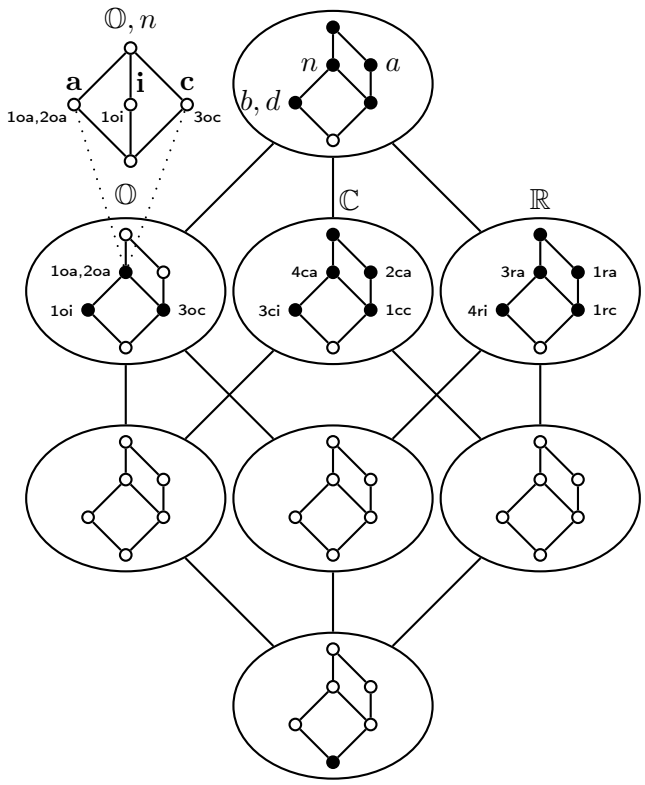


Figure 3.19: Summarized knowledge differentiated by subject role

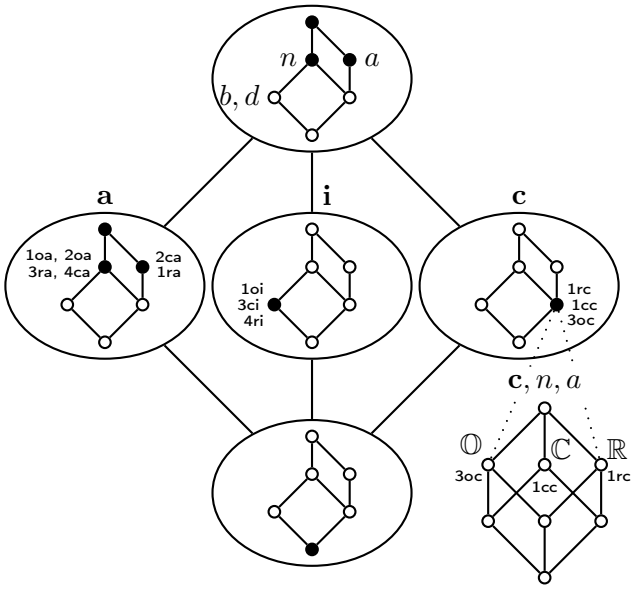


Figure 3.20: Summarized knowledge differentiated by purpose

	m	**i**	**c**
m	×	×	×
i		×	
c			×

m – medical treatment
i – infection treatment
c – cancer treatment

Table 3.23: Purpose lattice

considered purposes as a context, confer Table 3.23. The relation represents which purposes are implied by a specific purpose. In our example, *medical treatment* implies *treatment of infections* and *cancer treatment*, in addition.

We, still, assume that only one purpose can be related to each message. Then, we can summarize all one-valued purpose attributes in one many-valued attribute where purposes are attribute values. This many-valued attribute can be scaled by a context like Table 3.23. The scaled context provides, then, not just the purpose of a message, but also all implied purposes for each message.

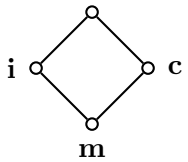


Figure 3.21: Purpose lattice

3.4.3 Obligations

Besides the purpose, the disclosure of personal data can be connected with obligations. They affect the recipient of data items. An obligation can be, for instance, the duty to delete data items after a specific period of time or notification of the data subject on specific events like disclosure to a third party or deletion.

Obligations are, like purposes, *attributes of messages*. Existing contexts can be enhanced by obligation attributes the same way as we showed for the context enhancement by purposes.

3.5 Supplementary knowledge

So far, we considered *data items* which are known from messages. This would be sufficient, if we observe each single message which has ever been sent between two subjects. Usually, this is not possible. We have to introduce a representation of *supplementary knowledge* to consider the knowledge which has not been covered by *observed* messages.

In general, every context which contains *data items* and *subjects* can be enhanced with supplementary knowledge. In case of *messages* as formal objects, we introduce another *type* of object which has only *subjects* and *data items* as attributes, i. e. we omit all attributes which are not known, for instance, about the source of information. In case of *subjects* as formal objects, we just have to add the *data item* attributes to the corresponding line in the context.

Contexts with tuples as formal objects can be enhanced in a similar way. If *messages*

belong to the object tuple, we have to proceed as described in the previous paragraph where contexts with *messages* as formal objects have been described. Otherwise, we have to proceed as described in the previous paragraph where contexts with *subjects* as formal objects have been described. However, it can happen that parts of an object tuple are not known. These parts have to be replaced by wild cards, i. e. yet unused symbols which are not meant to be reused, later.

3.6 Summary

In this chapter, we present our application approach of Formal Concept Analysis to communication contexts. We started with data contexts and corresponding lattices. They turned out to be very useful in following sections as formal scales to conclude and hide the complexity of different data item combinations and implications. Data types have been introduced to summarize different data items which provide similar data.

Messages contexts, as introduced in Section 3.2, with messages as formal objects and data items as formal attributes lead to the first concept lattice which can be used to find all similarities between messages with respect to data items which have been sent within these messages.

By taking connection control data into account, we refine the description of messages. They might, for instance, contain similar sets of data items, but have been sent to different subjects, as described in Section 3.3.1, or rather sent by different pseudonyms, as suggested in Section 3.3.2. However, subjects and pseudonyms are not necessarily independent from each other. As a matter of fact, a pseudonym needs to be created by some subject before it can be used for the first time. This relation leads to an additional formal scale for message contexts like the data context, previously.

Subjects, however, are more than attributes of messages. They can rather be sorted by known data items the same way like messages have been sorted by contained data

items, previously. This application is described in 3.3.3. These subject contexts can later be united with message contexts in order find similarities, and therefore links, between messages and subjects with respect to data items.

Both kinds of contexts can be enhanced by a separation of subjects in originators, recipients, and later in data subjects, too, described in Section 3.3.4 and 3.4.1. The latter separation belongs already to the communication context rather than to connection control data, as well as the purpose of a message, for instance. The context enhancement for message purposes is described in Section 3.4.2 and leads to a further formal scale, as also data items and pseudonyms did, previously. The same way can further attributes enhance these contexts, as pointed out in Section 3.3.5 and 3.4.3.

We do yet not take behavior, actions, and reactions into account. First of all, they do not provide further attributes which are useful to compare messages and subjects, since they are only attributes of subjects. Second, behavior consists of actions taken in sequence, but to compare action sequences, we need to model time. Easy solutions, like a counter, as it is used in many digital systems, for instance, is not suitable, however. Sequences which include two actions in sequence would, then, be completely different, if one of these sequences include a third action right in between the two similar actions. Third, it is yet not clear that recorded and analyzed behavior would lead to results which are useable for analyzing linkability. Similar behavior in particular situations is rather expectable, even if communication takes place with very different communication partners.

Chapter 4

Application Scenario

In this chapter, we use the *E-Shopping* scenario from [ACC+05] to construct situations where a customer has to decide about revealing personal data. We use different communication partners, payment and delivery conditions, and disclosed data items in several iteration steps and look at linkability between messages, between customer pseudonyms, and between messages and pseudonyms. We assume that different communication partners do not work together.

From [ACC+05] we take the separation of each E-Shopping process in three phases, *browsing*, *negotiation and purchase*, *payment*, and *delivery*. In the first phase, a customer has exactly one communication partner, the merchant, and reveals only his interest in a specific product and perhaps the fact of participating in a merchant's loyalty program. The customer uses transaction pseudonyms in this phase and his browsing behavior can be kept secret, since different browser requests are, therefore, not linkable by pseudonyms. They need even not be linkable by message contents. Data items in such request messages are descriptors of goods provided by the merchant. Thus, they are already known to the merchant and not usable to link messages and customer subjects explicitly. An exception are customer specific offers, however. These requests may be linked by message contents, i. e. offer descriptors, to the customer or to one of his previously used pseudonyms.

In the second phase, *negotiation and purchase*, the communication is also limited to customer and merchant. However, the merchant may request personal customer data which is necessary to legitimate the purchase. Selling specific goods might be restricted by law to customers older than a certain age, for instance. Then, the customer has to prove that all these requirements are fulfilled and, therefore, has to reveal personal data. Customers may even switch to relationship pseudonyms, in this phase, or show that several transaction pseudonyms belong together to build up reputation with respect to a merchant. Thus, messages can become linkable in this phase by both, pseudonyms, in case the customer reuses previously used pseudonyms, and message contents, in case the customer provides personal data which has been provided before.

For *payment* in the third phase, the customer can provide a credit card number, but he has to use the same pseudonym which has been used in the second phase. Thus, all messages which belong to this purchase would become linkable to messages of other purchases, if the same credit card number has been used by the customer in these messages. As a matter of fact, all related pseudonyms and data items become, then, also linkable. The alternative are *anonymous e-coins* which are not linkable to each other and, therefore, do not link different messages by their occurrence. Anonymous e-coins can be implemented as described by Chaum in [CFN90] by anonymous credentials as described by Camenisch and Herreweghen in [CH02].

For *delivery*, we consider three opportunities, *direct delivery* by the merchant, delivery by a *third party service*, and anonymous delivery to a *pick-up point*. In each case, an address has to be revealed. However, it is usually not possible to choose different home addresses as frequently as choosing a new pick-up point, for instance, for each purchase. Thus, revealing the home address yields a higher likelihood of being linked by that data item. By using a delivery service, it is possible to separate the knowledge about the product from the knowledge about the recipient, as pointed out in [ACC+05]. The product is, then, known to the merchant, but not to the delivery service, whereas the customer's address is

known to the delivery service, but not to the merchant.

4.1 Scenario outline

In this scenario, we start from scratch. Yet, the customer has not registered any data item or any message to the formal contexts. However, he became curious about the release of his personal data in his daily live. He decides, therefore, to register every item when it is released.

After browsing through offers of an online book store, the customer decides to buy the online version of a scientific book. To his surprise, he gets a voucher, additionally to the ordered book. The bookstore offers him a certain discount on the paperback edition of the same book. He stores it for taking the decision later, especially, since there is also the note that the next revision of this book is already in work and is going to be published, soon.

When reading through the book, he learns about a software which seems to be very useful in his daily work. He takes a distributor which provides this software to fair conditions and decides to buy it. The software producer offers online support for all customers. Thus, when buying this software, the customer gets also a certificate which states that he legally bought it.

After reading the book and installing all software components the right way, the customer realizes that he forgot to go to the supermarket. It is too late for the ordinary market, but online orders still work and will be delivered in time.

Some days later, the customer has already some experiences with his new software and is used to the book as reference. He notices that the new book revision is already available and decides to buy the paperback edition, since his monitor seems to be to crowded by showing both, the online book and the software interface. He finds his voucher and gets the discount. In addition, the book store offers him to participate in a discount system. There, he can collect bonus points in order to get further discounts in future. He accepts,

since the system seems to be fair in a way that it demands further information just right in this moment when the customer demands discount.

Again, it is late and the customer decides to buy at an online supermarket. This time, he decides to choose one where he can pay anonymously, as it would be in an ordinary supermarket, and where the opportunity is provided to choose a pickup point which is on his way to work.

In the new revision of his book, the customer reads about software features which seems not to work in his installation. After a while, he decides to contact the support which helps to track down the problem.

4.2 E-book purchase

In the first part of the scenario, the customer browses anonymously through offers of his communication partner, the online book store. He uses different previously unused pseudonyms for each request. Nevertheless, he logs all requests, even if they seem to be unlinkable by pseudonyms. His browsing activity causes three messages, all refining his specific interest step by step. In his first message, he requests all books, but the store has quite a lot. Then, in the second message, he refines his request to cover all books about Computer Aided Design. Still, there are too many hits, so, the third message is, again, a refinement in order to receive all books about Computer Aided Design which have been published recently. Now, the list of available books is short enough to determine the one which fits his needs best.

The customer uses a new, i. e. previously unused, pseudonym to order the chosen book. In the payment negotiation, he notices the opportunity to pay with *e-coins*, but, for now, he decides to use his credit card, since he is used to credit card payment and has never heard about e-coins, before. However, he is curious about this alternative payment and decides to ask at his bank office for more information. The store requests, then, credit card

	i_b	i_c	i_r	o_b	c	b_v
i_1	×					
i_2	×	×				
i_3	×	×	×			
o_b	×	×	×	×		
c					×	
b_v	×	×	×	×		×

i_1, i_b – interested in books
i_2 – interested in books about CAD
i_c – interested in CAD
i_3 – interested in recently published books about CAD
i_r – interested in recently published items
o_b – order of the book *CAD in particular*
c – credit card number & expiration date
b_v – voucher

Table 4.1: Data context for e-book purchase

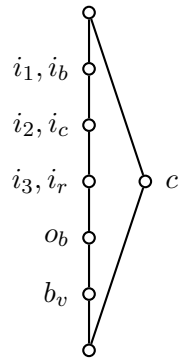


Figure 4.1: Data lattice for e-book purchase

number and the expiration date of this card.

As response, the customer finds a voucher for the paperback edition tacked to the ordered e-book. He registers even the voucher as data item in the data and message context.

We summarize the content of his data context in Table 4.1 and his message context in Table 4.2 and 4.3. The latter table consists of two parts, the first is the result of scaling Table 4.2 like described in *scaling role depending on message* in Section 3.3.4 and the second is the result of scaling like described in *pseudonyms, data items, and subject roles* in the same section. Table 4.3 unites both contexts in order to obtain relations between only messages and pseudonyms, in addition to relations from each specific context, i. e. between

	data item	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	purpose
m_{b01}	i_1	$\mathbb{P}_1$	$\mathbb{B}$	$\mathbb{P}_1$	**b**
m_{b02}	i_2	$\mathbb{P}_2$	$\mathbb{B}$	$\mathbb{P}_2$	**b**
m_{b03}	i_3	$\mathbb{P}_3$	$\mathbb{B}$	$\mathbb{P}_3$	**b**
m_{b04}	o_b	$\mathbb{P}_4$	$\mathbb{B}$	$\mathbb{P}_4$	**o**
m_{b05}	c	$\mathbb{P}_4$	$\mathbb{B}$	$\mathbb{P}_4$	**p**
m_{b06}	b_v	$\mathbb{B}$	$\mathbb{P}_4$	$\mathbb{P}_4$	**d**

i_1 – interested in books
i_2 – interested in books about CAD
i_3 – interested in recently published books about CAD
o_b – order of the book *CAD in particular*
c – credit card number & expiration date
b_v – voucher
$m_{b01}, \ldots, m_{b06}$ – messages
$\mathbb{P}_1, \ldots, \mathbb{P}_4$ – customer's pseudonyms
$\mathbb{B}$ – book store pseudonym
b – purpose: browsing
o – purpose: order
p – purpose: payment
d – purpose: delivery

Table 4.2: Message context for e-book purchase

	i_b	i_c	i_r	o_b	c	b_v	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	label
$(m_{b01}, \mathbb{P}_1)$	×						×		×	mb11
$(m_{b01}, \mathbb{B})$	×							×		mb1b
$(m_{b02}, \mathbb{P}_2)$	×	×					×		×	mb22
$(m_{b02}, \mathbb{B})$	×	×						×		mb2b
$(m_{b03}, \mathbb{P}_3)$	×	×	×				×		×	mb33
$(m_{b03}, \mathbb{B})$	×	×	×					×		mb3b
$(m_{b04}, \mathbb{P}_4)$	×	×	×	×			×		×	mb44
$(m_{b04}, \mathbb{B})$	×	×	×	×				×		mb4b
$(m_{b05}, \mathbb{P}_4)$					×		×		×	mb54
$(m_{b05}, \mathbb{B})$					×			×		mb5b
$(m_{b06}, \mathbb{P}_4)$	×	×	×	×		×		×	×	mb64
$(m_{b06}, \mathbb{B})$	×	×	×	×		×	×			mb6b
$(\mathbb{P}_1, \mathbb{O})$	×						×			p1o
$(\mathbb{P}_1, \mathbb{C})$	×								×	p1c
$(\mathbb{P}_2, \mathbb{O})$	×	×					×			p2o
$(\mathbb{P}_2, \mathbb{C})$	×	×							×	p2c
$(\mathbb{P}_3, \mathbb{O})$	×	×	×				×			p3o
$(\mathbb{P}_3, \mathbb{C})$	×	×	×						×	p3c
$(\mathbb{P}_4, \mathbb{O})$	×	×	×	×	×		×			p4o
$(\mathbb{P}_4, \mathbb{R})$	×	×	×	×		×		×		p4r
$(\mathbb{P}_4, \mathbb{C})$	×	×	×	×	×	×			×	p4c
$(\mathbb{B}, \mathbb{O})$	×	×	×	×		×	×			pbo
$(\mathbb{B}, \mathbb{R})$	×	×	×	×	×			×		pbr

Table 4.3: Scaled message context for e-book purchase

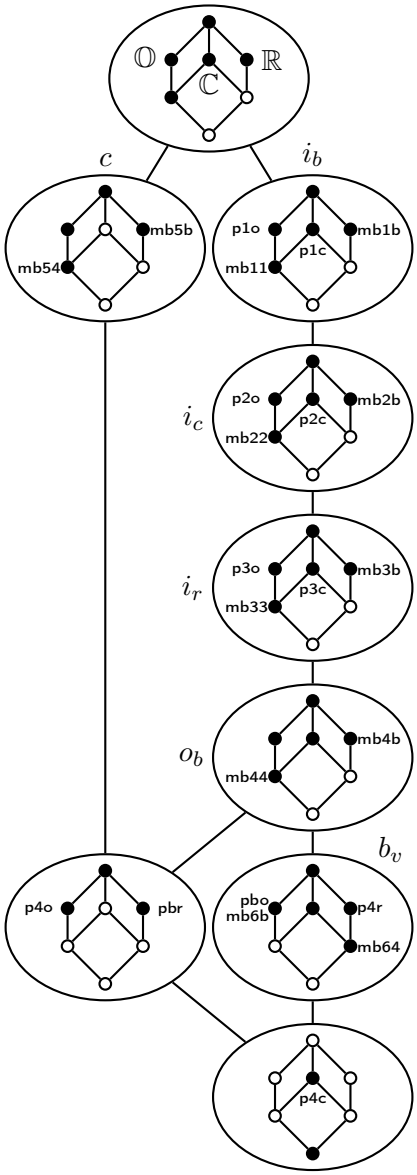


Figure 4.2: Nested message lattice for e-book purchase

messages resp. pseudonyms.

The credit card number is unique in the world and, therefore, best qualified for tracing a subject, even if different pseudonyms are used. The customer chooses this data item as a starting point for a first linkability analysis. First, he wants to determine all of his pseudonyms where his credit card number is known to be related to. Therefore, he chooses the set of formal attributes $\{\mathbb{C}, c\}$ and computes the first derivation in his message context.

$$\{\mathbb{C}, c\}' = \{(\mathbb{P}_4, \mathbb{C}), (m_{b05}, \mathbb{P}_4)\}$$

He computes, then, all data items which are known of his pseudonym $\mathbb{P}_4$. The computation leads to the concept (A'', A') with $A = \{(\mathbb{P}_4, \mathbb{C})\}$. It is labeled as p4c in Figure 4.2.

$$\begin{aligned}\{(\mathbb{P}_4, \mathbb{C})\}' &= \{i_b, i_c, i_r, o_b, c, b_v, \mathbb{C}\} \\ \{(\mathbb{P}_4, \mathbb{C})\}'' &= \{i_b, i_c, i_r, o_b, c, b_v, \mathbb{C}\}' \\ &= \{(\mathbb{P}_4, \mathbb{C})\}\end{aligned}$$

One of the ancestors of (A'', A') within the concept lattice is labeled as p3c in Figure 4.2 and formally

$$\Big(\{(m_{b03}, \mathbb{P}_3), (m_{b04}, \mathbb{P}_4), (m_{b06}, \mathbb{P}_4), (\mathbb{P}_3, \mathbb{C}), (\mathbb{P}_4, \mathbb{C})\}, \{i_b, i_c, i_r, \mathbb{C}\}\Big)$$

The customer concludes that $\mathbb{P}_3$, $\mathbb{P}_4$, m_{b03}, m_{b_04}, and m_{b_06} are related to each other by a relatively large amount of data items. It is not surprising in case of $\mathbb{P}_4$ and m_{b04} resp. m_{b06}, since both messages reveal data about this pseudonym. But, in contrast to this obvious relation, the relation between $\mathbb{P}_4$ and $\mathbb{P}_3$ resp. m_{b03} is only caused by similar data item sets. Drawing the full circle, it can be concluded that $\mathbb{P}_3$ is related by these data items to the

credit card number, even though it was only used for browsing.

4.3 Software purchase

When the customer is looking for the software which has been mentioned in his new book, he knows, in contrast to the previously described book purchase, exactly what he wants. Therefore, he does not need to browse through several offers of one distributor, but can request specific offers from different software distributors. He uses, again, a yet unused pseudonym for each request to avoid linkability by pseudonyms.

The customer decides, then, to order at one of these distributor sites and reuses the pseudonym which he had used before in his request, since he wants to buy the software under the same conditions which have already been proposed. After consulting his bank office, he decides, in addition, to use *anonymous e-coins*, rather than credit card payment. He wants also to avoid linkability by his credit card number which he, otherwise, would have to reveal again.

When the software arrives, the customer notices an additional certificate which has been delivered together with the software. It states that he bought the software and a corresponding timestamp is included. He is, therewith, permitted to make use of the software producer's support division and to receive bug fixes for his software version.

The entire purchase is described formally in Table 4.4 and 4.5. The customer adds both tables to his message and data context, respectively. The scaled message context is presented in Table 4.6 which also contains the context of Table 4.3 on page 70. It is, therefore, the complete message context which is known to the customer, so far.

Two pseudonyms have been used for orders at all. The customer wants to know the pseudonyms exactly and get an idea of how linkable they are by his disclosed data items.

	data item	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	purpose
m_{s01}	i_4	$\mathbb{P}_5$	$\mathbb{D}_1$	$\mathbb{P}_5$	**b**
m_{s02}	i_4	$\mathbb{P}_6$	$\mathbb{D}_2$	$\mathbb{P}_6$	**b**
m_{s03}	i_4	$\mathbb{P}_7$	$\mathbb{D}_3$	$\mathbb{P}_7$	**b**
m_{s04}	o_s	$\mathbb{P}_5$	$\mathbb{D}_1$	$\mathbb{P}_5$	**o**
m_{s05}	c_1	$\mathbb{P}_5$	$\mathbb{D}_1$	$\mathbb{P}_5$	**p**
m_{s06}	s_l	$\mathbb{D}_1$	$\mathbb{P}_5$	$\mathbb{P}_5$	**d**

i_4 – interested in software *CAD in particular*
o_s – order of the software
c_1 – anonymous e-coins
s_l – software legitimation
$m_{s01}, \ldots, m_{s06}$ – messages
$\mathbb{P}_5, \ldots, \mathbb{P}_7$ – customer's pseudonyms
$\mathbb{D}_1$ – software distributor pseudonym
b – purpose: browsing
o – purpose: order
p – purpose: payment
d – purpose: delivery

Table 4.4: Message context for software purchase

	i_c	i_s	o_s	c_1	s_l
i_4	×	×			
o_s	×	×	×		
c_1				×	
s_l	×	×	×		×

i_4 – interested in software *CAD in particular*
i_c – interested in CAD
i_s – interested in software
o_s – order of the software
c_1 – anonymous e-coins
s_l – software legitimation

Table 4.5: Data context for software purchase

	i_b	i_c	i_r	i_s	o_b	o_s	c	c_1	b_v	s_l	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	label
$(m_{b01}, \mathbb{P}_1)$	×										×		×	mb11
$(m_{b01}, \mathbb{B})$	×											×		mb1b
$(m_{b02}, \mathbb{P}_2)$	×	×									×		×	mb22
$(m_{b02}, \mathbb{B})$	×	×										×		mb2b
$(m_{b03}, \mathbb{P}_3)$	×	×	×								×		×	mb33
$(m_{b03}, \mathbb{B})$	×	×	×									×		mb3b
$(m_{b04}, \mathbb{P}_4)$	×	×	×		×						×		×	mb44
$(m_{b04}, \mathbb{B})$	×	×	×		×							×		mb4b
$(m_{b05}, \mathbb{P}_4)$							×				×		×	mb54
$(m_{b05}, \mathbb{B})$							×					×		mb5b
$(m_{b06}, \mathbb{P}_4)$	×	×	×		×				×			×	×	mb64
$(m_{b06}, \mathbb{B})$	×	×	×		×				×		×			mb6b
$(m_{s01}, \mathbb{P}_5)$		×		×							×		×	ms15
$(m_{s01}, \mathbb{D}_1)$		×		×								×		ms1d1
$(m_{s02}, \mathbb{P}_6)$		×		×							×		×	ms26
$(m_{s02}, \mathbb{D}_2)$		×		×								×		ms2d2
$(m_{s03}, \mathbb{P}_7)$		×		×							×		×	ms37
$(m_{s03}, \mathbb{D}_3)$		×		×								×		ms3d3
$(m_{s04}, \mathbb{P}_5)$		×		×		×					×		×	ms45
$(m_{s04}, \mathbb{D}_1)$		×		×		×						×		ms4d1
$(m_{s05}, \mathbb{P}_5)$								×			×		×	ms55
$(m_{s05}, \mathbb{D}_1)$								×				×		ms5d1
$(m_{s06}, \mathbb{P}_5)$		×		×		×				×		×	×	ms65
$(m_{s06}, \mathbb{D}_1)$		×		×		×				×	×			ms6d1
$(\mathbb{P}_1, \mathbb{O})$	×										×			p1o
$(\mathbb{P}_1, \mathbb{C})$	×												×	p1c
$(\mathbb{P}_2, \mathbb{O})$	×	×									×			p2o
$(\mathbb{P}_2, \mathbb{C})$	×	×											×	p2c
$(\mathbb{P}_3, \mathbb{O})$	×	×	×								×			p3o
$(\mathbb{P}_3, \mathbb{C})$	×	×	×										×	p3c
$(\mathbb{P}_4, \mathbb{O})$	×	×	×		×		×				×			p4o
$(\mathbb{P}_4, \mathbb{R})$	×	×	×		×				×			×		p4r
$(\mathbb{P}_4, \mathbb{C})$	×	×	×		×		×		×				×	p4c
$(\mathbb{B}, \mathbb{O})$	×	×	×		×				×		×			pbo
$(\mathbb{B}, \mathbb{R})$	×	×	×		×		×					×		pbr
$(\mathbb{P}_5, \mathbb{O})$		×		×		×		×			×			p5o
$(\mathbb{P}_5, \mathbb{R})$		×		×		×				×		×		p5r
$(\mathbb{P}_5, \mathbb{C})$		×		×		×		×		×			×	p5c
$(\mathbb{P}_6, \mathbb{O})$		×		×							×			p6o
$(\mathbb{P}_6, \mathbb{C})$		×		×									×	p6c
$(\mathbb{P}_7, \mathbb{O})$		×		×							×			p7o
$(\mathbb{P}_7, \mathbb{C})$		×		×									×	p7c
$(\mathbb{D}_1, \mathbb{O})$		×		×		×				×	×			pd1o
$(\mathbb{D}_1, \mathbb{R})$		×		×		×		×				×		pd1r
$(\mathbb{D}_2, \mathbb{R})$		×		×								×		pd2r
$(\mathbb{D}_3, \mathbb{R})$		×		×								×		pd3r

Table 4.6: Scaled message context after software purchase; same symbols as in previous tables have been used

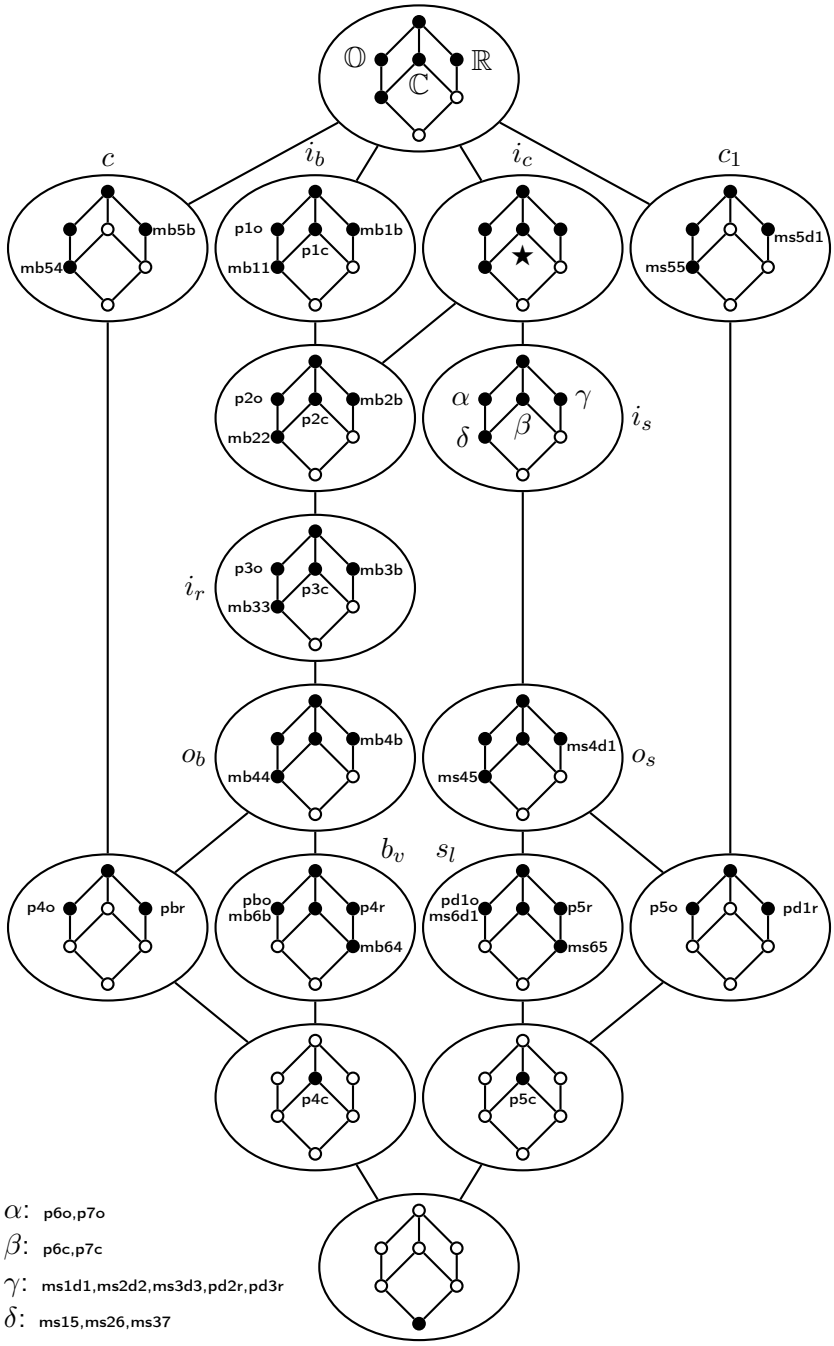


Figure 4.3: Nested message lattice after software purchase

He computes, therefore, the first derivations of all sets which contain already sent orders.

$$\{o_b, \mathbb{O}\}' = \{(m_{b04}, \mathbb{P}_4), (m_{b06}, \mathbb{B}), (\mathbb{P}_4, \mathbb{O}), (\mathbb{B}, \mathbb{O})\}$$
$$\{o_s, \mathbb{O}\}' = \{(m_{s04}, \mathbb{P}_5), (m_{s06}, \mathbb{D}_1), (\mathbb{P}_5, \mathbb{O}), (\mathbb{D}_1, \mathbb{O})\}$$

His own pseudonyms are only $\mathbb{P}_4$ and $\mathbb{P}_5$, but neither $\mathbb{B}$ nor $\mathbb{D}_1$, so he computes the derivation of $\{(\mathbb{P}_4, \mathbb{C}), (\mathbb{P}_5, \mathbb{C})\}$ to obtain all data items which are known of those both pseudonyms.

$$\{(\mathbb{P}_4, \mathbb{C}), (\mathbb{P}_5, \mathbb{C})\}' = \{i_c, \mathbb{C}\}$$

He notices, just the fact that he is interested in Computer Aided Design is a link between both purchases. Then, he wonders if this data item is also known of other pseudonyms and computes the entire concept (B', B'') where B is chosen as $\{i_c, \mathbb{C}\}$. This concept is labeled as ★ in Figure 4.3.

$$\begin{aligned}(B', B'') = \Big(\big\{&(\mathbb{P}_2, \mathbb{C}), (\mathbb{P}_3, \mathbb{C}), (\mathbb{P}_4, \mathbb{C}), (\mathbb{P}_5, \mathbb{C}), (\mathbb{P}_6, \mathbb{C}), (\mathbb{P}_7, \mathbb{C}),\\ &(m_{b02}, \mathbb{P}_2), (m_{b03}, \mathbb{P}_3), (m_{b04}, \mathbb{P}_4), (m_{b06}, \mathbb{P}_4), (m_{s01}, \mathbb{P}_5),\\ &(m_{s02}, \mathbb{P}_6), (m_{s03}, \mathbb{P}_7), (m_{s04}, \mathbb{P}_5), (m_{s06}, \mathbb{P}_5)\big\}, \{i_c, \mathbb{C}\}\Big)\end{aligned}$$

So, we see that this data item is known about almost all pseudonyms in the message context.

4.4 Online supermarket

The customer is in a hurry when he enters the supermarket site. Only one market is still open, so he has no chance to choose another communication partner, since he has to buy in time. He chooses directly goods and accepts to pay by credit card, since the market

	data item	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	purpose
m_{m001}	i_5	$\mathbb{P}_8$	$\mathbb{M}_1$	$\mathbb{P}_8$	**b**
m_{m002}	o_{m1}	$\mathbb{P}_8$	$\mathbb{M}_1$	$\mathbb{P}_8$	**o**
m_{m003}	c	$\mathbb{P}_8$	$\mathbb{M}_1$	$\mathbb{P}_8$	**p**
m_{m004}	a	$\mathbb{P}_8$	$\mathbb{M}_1$	$\mathbb{P}_8$	**d**
m_{m005}	i_6	$\mathbb{P}_9$	$\mathbb{M}_1$	$\mathbb{P}_9$	**b**
m_{m006}	o_{m2}	$\mathbb{P}_9$	$\mathbb{M}_1$	$\mathbb{P}_9$	**o**
m_{m007}	c	$\mathbb{P}_9$	$\mathbb{M}_1$	$\mathbb{P}_9$	**p**
m_{m008}	a	$\mathbb{P}_9$	$\mathbb{M}_1$	$\mathbb{P}_9$	**d**
m_{m009}	i_7	$\mathbb{P}_{10}$	$\mathbb{M}_1$	$\mathbb{P}_{10}$	**b**
m_{m010}	o_{m3}	$\mathbb{P}_{10}$	$\mathbb{M}_1$	$\mathbb{P}_{10}$	**o**
m_{m011}	c	$\mathbb{P}_{10}$	$\mathbb{M}_1$	$\mathbb{P}_{10}$	**p**
m_{m012}	a	$\mathbb{P}_{10}$	$\mathbb{M}_1$	$\mathbb{P}_{10}$	**d**

i_5 – interested in eggs
i_6 – interested in flour
i_7 – interested in milk
a – home address
c – credit card number & expiration date
o_{m01} – order of eggs
o_{m02} – order of flour
o_{m03} – order of milk
$m_{m001}, \ldots, m_{m012}$ – messages
$\mathbb{P}_8, \ldots, \mathbb{P}_{10}$ – customer's pseudonyms
$\mathbb{M}_1$ – supermarket pseudonym
b – purpose: browsing
o – purpose: order
p – purpose: payment
d – purpose: delivery

Table 4.7: Message context for supermarket purchase

does not support payment by e-coins.

The market provides shopping carts which are realized as session pseudonyms. A session is meant to last for one purchase. However, the customer feels uncomfortable because of his lack of time and options. Shopping carts may lead to customer profiles and, therefore, to disclosure of personal data which can hardly be controlled afterwards. So, he decides to avoid shopping carts as they are intended by the market site and buys each product in a separate cart, i. e. he uses several previously unused pseudonyms. In addition, the customer has to provide a delivery address, since only *direct delivery* is supported by the market.

The messages which were necessary for this purchase can be found in Table 4.7 and the corresponding scale in Table 4.8.

The scaled context can be computed as shown in the last sections. We use the reference

	i_e	i_f	i_m	a	c	o_{m01}	o_{m02}	o_{m03}
i_5	×							
i_6		×						
i_7			×					
a				×				
c					×			
o_{m01}	×					×		
o_{m02}		×					×	
o_{m03}			×					×

i_5, i_e – interested in eggs
i_6, i_f – interested in flour
i_7, i_m – interested in milk
a – home address
c – credit card number & expiration date
o_{m01} – order of eggs
o_{m02} – order of flour
o_{m03} – order of milk

Table 4.8: Data context for supermarket purchase

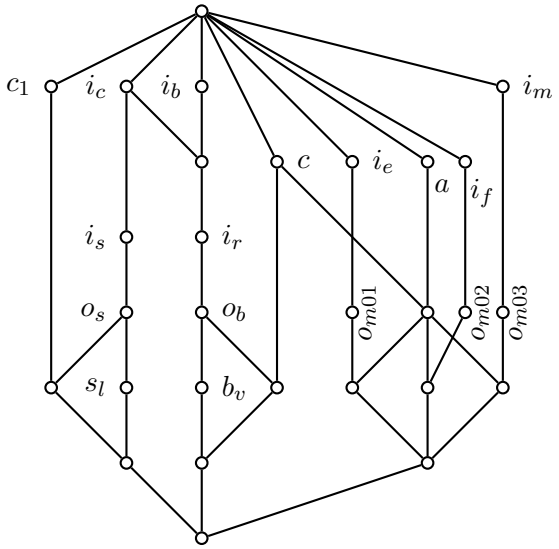


Figure 4.4: Sketch of the outer message lattice structure after visiting the market

implementation which we introduce in Chapter 5 and list in Appendix A. The source code for the scaled context can be found in Appendix B as *context data function* **tabScenMarket**. It can be used to reproduce the following computations.

One major interest of the customer was to avoid linkable shopping carts. Unfortunately, the carts are linkable by his credit card number and the delivery address. We can show this by computing the first derivation of the set of all shopping carts resp. the used pseudonyms.

$$\{(\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_9, \mathbb{C}), (\mathbb{P}_{10}, \mathbb{C})\}' = \{a, c, \mathbb{C}\}$$
$$\{(\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_9, \mathbb{C}), (\mathbb{P}_{10}, \mathbb{C})\}'' = \{(\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_9, \mathbb{C}), (\mathbb{P}_{10}, \mathbb{C})\}$$

Another important result can be obtained by traversing upwards through the lattice structure. One upper neighbor of the previously computed concept ($\{(\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_9, \mathbb{C}), (\mathbb{P}_{10}, \mathbb{C})\}, \{a, c, \mathbb{C}\}$) reveals the linkability between the purchase in the supermarket and the e-book purchase which had been done using $\mathbb{P}_4$.

$$\Big(\{(\mathbb{P}_4, \mathbb{C}), (\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_9, \mathbb{C}), (\mathbb{P}_{10}, \mathbb{C}), (m_{b05}, \mathbb{P}_4), (m_{m003}, \mathbb{P}_8), (m_{m007}, \mathbb{P}_9), (m_{m011}, P_{11})\}, \{\mathbb{C}, c\}\Big)$$

4.5 Paperback edition purchase

The new revision of the previously bought book has been published and the customer decides to use his voucher to get discount on the purchase. He selects the new edition on the book store site and goes straight to pay it. This time, he is going to use e-coins. In addition, he has to provide a delivery address, since the paperback edition can hardly be delivered online.

The store supports *direct delivery*, *delivery through a third party*, or *delivery to a pickup-point*. The customer chooses the cheapest option, direct delivery in this case, out of his

	data item	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	purpose
m_{b11}	o_{bp}	$\mathbb{P}_{11}$	$\mathbb{B}$	$\mathbb{P}_{11}$	**o**
m_{b12}	b_v	$\mathbb{P}_{11}$	$\mathbb{B}$	$\mathbb{P}_{11}$	**p**
m_{b13}	a	$\mathbb{P}_{11}$	$\mathbb{B}$	$\mathbb{P}_{11}$	**d**
m_{b14}	c_2	$\mathbb{P}_{11}$	$\mathbb{B}$	$\mathbb{P}_{11}$	**p**
m_{b15}	b_b	$\mathbb{B}$	$\mathbb{P}_{11}$	$\mathbb{P}_{11}$	**p**

o_{bp} – order of the book
b_v – voucher
a – home address
c_2 – e-coins
b_b – bonus point
$m_{b11}, \ldots, m_{b15}$ – messages
$\mathbb{P}_{11}$ – customer's pseudonym
$\mathbb{B}$ – supermarket pseudonym
o – purpose: order
p – purpose: payment
d – purpose: delivery

Table 4.9: Message context for book purchase

confusion about the unexpected choice.

After processing the book order, the store sends an additional certificate. It includes value and date of purchase and is meant to be revealed as bonus point by the customer, for instance, to get further discount on the next purchase at this store.

For the entire purchase, only one pseudonym is necessary, since the customer does not browse on the book store site and linkability between the payment and delivery negotiation steps is needed to assign the sold product to the right delivery address. The linkability to other purchases can be simulated in advance, since proper constraints are specified between customer and book store during the purchase negotiation step. The corresponding source code can be found in Appendix B as **tabScenPaper**.

The customer adds the corresponding messages, shown in Table 4.9, to the message context and scales them for his simulation. For the scaling, he uses his data context enhanced by Table 4.10. The first book purchase is linkable to the actual purchase by the issued voucher, his interests in books, CAD, and recently published items. Additionally, the actual purchase is linkable to the supermarket purchase by the customer's home address. Both facts can be concluded from computing the upper neighbors of (A'', A') where $A =$

	i_b	i_c	i_r	o_b	o_{bp}	c_2	a	b_v	b_b
o_{bp}	×	×	×		×				
c_2						×			
a							×		
b_v	×	×	×	×				×	
b_b									×

i_b – interested in books
i_c – interested in CAD
i_r – interested in recently published items
o_b – order of the book
o_b – order of the paperback edition
c_2 – e-coins
a – address
b_v – voucher
b_b – bonus point

Table 4.10: Data context for e-book purchase

$\{(\mathbb{P}_{11}, \mathbb{C})\}$. The two important neighbors are

$$\Big(\big\{(\mathbb{P}_4, \mathbb{C}), (\mathbb{P}_{11}, \mathbb{C}), (m_{b06}, \mathbb{P}_4), (m_{b12}, \mathbb{P}_{11})\big\}, \big\{\mathbb{C}, b_v, i_b, i_c, i_r, o_b\big\}\Big)$$

$$\Big(\big\{(\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_9, \mathbb{C}), (\mathbb{P}_{10}, \mathbb{C}), (\mathbb{P}_{11}, \mathbb{C}), (m_{b13}, \mathbb{P}_{11}),$$
$$(m_{m004}, \mathbb{P}_8), (m_{m008}, \mathbb{P}_9), (m_{m012}, \mathbb{P}_{10})\big\}, \big\{\mathbb{C}, a\big\}\Big)$$

The customer accepts this linkability and commits his book order.

4.6 Another online supermarket

This time, the chosen supermarket arranged all the offers very clearly, so, the customer chooses directly all the goods and go to pay. Delivery to a pickup-point is supported.

Again, only one pseudonym is necessary for the entire purchase session. A simulation, however, is not supported by the shop. The customer registers the messages in Table 4.11 one after another and checks the linkability of each message to pseudonyms and messages which have been previously registered. The corresponding source code can be found in Appendix B as **tabScenMarketII**.

Right with the first message, the customer notices the linkability by similar interests

	data item	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	purpose
m_{m11}	o_{m1}	$\mathbb{P}_{12}$	$\mathbb{M}_2$	$\mathbb{P}_{12}$	**o**
m_{m12}	c_3	$\mathbb{P}_{12}$	$\mathbb{M}_2$	$\mathbb{P}_{12}$	**p**
m_{m13}	a_1	$\mathbb{P}_{12}$	$\mathbb{M}_2$	$\mathbb{P}_{12}$	**d**

o_{m1} – order of eggs, milk, and cinnamon
c_3 – e-coins
a_1 – pick-up point address
$m_{m11}, \ldots, m_{m13}$ – messages
$\mathbb{P}_{12}$ – customer's pseudonym
$\mathbb{M}_2$ – supermarket pseudonym
o – purpose: order
p – purpose: payment
d – purpose: delivery

Table 4.11: Message context for second supermarket purchase

	i_e	i_x	i_m	a_1	c_3	o_{m1}
a_1				×		
c_3					×	
o_{m1}	×	×	×			×

i_e – interested in eggs
i_m – interested in milk
i_x – interested in cinnamon
a_1 – pickup point address
c_3 – e-coins
o_{m1} – order of eggs, milk, and cinnamon

Table 4.12: Data context for second supermarket purchase

to his previous supermarket purchase. He computes the formal concept (A'', A') with $A = \{(m_{m11}, \mathbb{P}_{12})\}$ and traverses the lattice upwards.

$$\big(A'', A'\big) = \Big(\big\{(m_{m11}, \mathbb{P}_{12})\big\}, \big\{i_e, i_x, i_m, o_{m1}, \mathbb{O}, \mathbb{C}\big\}\Big)$$

Two of the upper neighbors are

$$\Big(\big\{(m_{m001}, \mathbb{P}_8), (m_{m002}, \mathbb{P}_8), (m_{m11}, \mathbb{P}_{12})\big\}, \big\{\mathbb{C}, \mathbb{O}, i_e\big\}\Big)$$
$$\Big(\big\{(m_{m009}, \mathbb{P}_{10}), (m_{m010}, \mathbb{P}_{10}), (m_{m11}, \mathbb{P}_{12})\big\}, \big\{\mathbb{C}, \mathbb{O}, i_m\big\}\Big)$$

He accepts this and checks how to pay. Payment by credit card, however, would make his actual purchase pseudonym linkable to several previous purchases, as the following

computation shows.

$$\{c, \mathbb{C}\}' = \{(\mathbb{P}_4, \mathbb{C}), (\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_9, \mathbb{C}), (\mathbb{P}_{10}, \mathbb{C}), \\ (m_{b05}, \mathbb{P}_4), (m_{m003}, \mathbb{P}_8), (m_{m007}, \mathbb{P}_9), (m_{m011}, \mathbb{P}_{10})\}$$

Therefore, he chooses payment by e-coins to avoid these unnecessary links. When choosing the delivery address, the customer notices, in addition, that he used his home address several times before, as shown in the next computation. The disclosure of it would, therefore, provide a further link to previous purchases.

$$\{a, \mathbb{C}\}' = \{(\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_9, \mathbb{C}), (\mathbb{P}_{10}, \mathbb{C}), (\mathbb{P}_{11}, \mathbb{C}), (m_{b13}, \mathbb{P}_{11}), \\ (m_{m004}, \mathbb{P}_8), (m_{m008}, \mathbb{P}_9), (m_{m012}, \mathbb{P}_{10})\}$$

Also, he likes the idea of picking up the goods on the way from his working place. So, he chooses a pickup point as delivery address. This purchase is, therefore, only linkable to the previous supermarket purchase, since second degree ancestors of (A'', A') with $A = \{(\mathbb{P}_{12}, \mathbb{C})\}$ include

$$\Big(\{(\mathbb{P}_{10}, \mathbb{C}), (\mathbb{P}_{12}, \mathbb{C}), (m_{m009}, \mathbb{P}_{10}), (m_{m010}, \mathbb{P}_{10}), (m_{m11}, \mathbb{P}_{12})\}, \{\mathbb{C}, i_m\}\Big) \\ \Big(\{(\mathbb{P}_8, \mathbb{C}), (\mathbb{P}_{12}, \mathbb{C}), (m_{m001}, \mathbb{P}_8), (m_{m002}, \mathbb{P}_8), (m_{m11}, \mathbb{P}_{12})\}, \{\mathbb{C}, i_e\}\Big)$$

4.7 Software support

Encouraged by hints of the new book revision, the customer discovers a flaw in his software. He calls the support to ask for help or a software update. The flaw is known and a software update available, but any further service is refused by the support division as long as the customer has not proved to be a legal user of this software product. Thus, the customer switches back to the pseudonym which he used to buy the software and shows his certificate

	data item	$\mathbb{O}$	$\mathbb{R}$	$\mathbb{C}$	purpose
m_{s11}	s_b	$\mathbb{P}_{13}$	$\mathbb{X}$	$\mathbb{P}_{13}$	**v**
m_{s12}	o_u	$\mathbb{P}_{14}$	$\mathbb{X}$	$\mathbb{P}_{14}$	**o**
m_{s13}	o_u	$\mathbb{P}_5$	$\mathbb{X}$	$\mathbb{P}_5$	**o**
m_{s14}	s_l	$\mathbb{P}_5$	$\mathbb{X}$	$\mathbb{P}_5$	**p**

s_b – bug report
o_u – software update request
s_l – software legitimation
$m_{s11}, \ldots, m_{s13}$ – messages
$\mathbb{P}_5, \mathbb{P}_{13}, \mathbb{P}_{14}$ – customer's pseudonyms
$\mathbb{X}$ – software producer pseudonym
v – purpose: verification
o – purpose: order

Table 4.13: Message context for software support request

	i_c	i_s	o_s	o_u	s_l
s_b	×	×			
o_u	×	×		×	
s_l	×	×	×		×

i_c – interested in CAD
i_s – interested in software
s_b – bug report
o_s – order of the software
o_u – software update request
s_l – software legitimation

Table 4.14: Data context for software support request

which has been issued by the software producer. Then, the software update is delivered immediately.

Additional messages which have been caused by this support request are listed in Table 4.13.

In this case, the software producer uses reputation to authenticate a customer for a service. The customer built up reputation by buying the software. In this case, the permission for software updates is only granted when a valid software purchase certificate can be shown as reputation.

There are links to the software purchase by the customer's interests in Computer Aided Design and software, and also by the pseudonym which has been used for the software purchase. Furthermore, there are also links between support request and both book purchases by the interest in Computer Aided Design. The corresponding concepts can be determined using the context data provided by function **tabScenSupport** in Appendix B.

4.8 Conclusions

We have seen that the context definitions which we introduce in Chapter 3 are very good suited for examination of similarities between messages and subjects within a communication system. Our scenario focuses on similarities between data item sets which characterize messages or pseudonyms. Message contents which have yet not been sent can be compared to revealed and, thus, known data items of a pseudonym or the contents of other messages, respectively, in order to choose the contents with lowest or highest correlation. However, these context definitions yield no opportunity to distinguish data items which identify subjects unambiguously from those which are widely used and are, therefore, ambiguous with respect to subjects. Thus, the credit card number appears as the same kind of data item as the interest in milk, whereas the credit card number would lead to much higher risk of re-identification, indeed.

Chapter 5

Implementation of Concept Lattices and its Complexity

So far, we discussed how Formal Concept Analysis can be applied to communication relations. In this chapter, we introduce operations to compute concept lattices or parts of lattices and take a look at their time complexity. We provide reference implementations in Haskell, a functional programming language, building on the common Haskell interpreter Hugs. We use the Hugs release of March 2005, because of its support for state monads which allows to design functions that run and can be combined within the same formal context. Haskell and its fundamental concepts, including monads, are described in detail in [Tho99].

5.1 Data structures

There are three fundamental data structures which are necessary for Formal Concept Analysis, context, concept, and lattice.

A *context* consists of a set of objects, a set of attributes, and an incidence relation, confer Definition 1. Sets can be declared by standard Haskell data types. Relations, however, can

be implemented in different ways. We decide to use two maps so that one maps objects to the set of related attributes and the other one does the reverse mapping from a given attribute to the set of related objects. This yields also an easy opportunity to represent many-valued contexts with a mapping $m(g) = w$, confer Definition 5, since $m(g) = w$ is just an enhancement of the reverse mapping of the incidence relation.

In addition, we know that the incidence relation alone is a sufficient definition for a formal context. We provide, therefore, the user interface type **Incidence** to create contexts initially, but relay computations on **Context** which provides a full qualified context.

Listing 5.1: Context data type (Concept)

```
data Context o a v =
65  Context
      (Set o)                    -- object set
67    (Set a)                    -- attribute set
      (Map o (Set a))            -- incidence relation
69    (Map a (Map o (Maybe v))) -- value assignment (and reverse incidence)
```

Concepts consist of a concept extent and a corresponding intent. Both are sets, extents consist of objects and intents of attributes. The definition is straight forward.

Listing 5.2: Concept data type (Concept)

```
81 data Concept o a = Concept (Set o) (Set a)
```

According to Definition 4, the *concept lattice* is the set of all concepts belonging to a *context*. However, Lindig pointed out in [Lin00] that most applications need to use the lattice structure, in addition to the concept set. We support this need by defining a triple for each concept in the lattice. The first component consists of the concept itself and the latter two components store upper and lower neighbors. We use a mapping in Haskell from concepts to these triples, since the data type **Map** is internally implemented as a balanced binary search tree.

Listing 5.3: Lattice data type (Concept)

```
data Lattice o a = Lattice (Map (Concept o a) (
  Concept o a,              -- concept
  Set (Concept o a),        -- upper neighbors
  Set (Concept o a)))       -- lower neighbors
```

5.2 Operations

The base operations of Formal Concept Analysis are *derivations* of an object set or an attribute set. They are necessary to compute a single concept from given objects or attributes. Then, we can arrange this concept in the concept lattice. The position of a concept within a concept lattice is determined by its neighbor concepts. *Upper neighbors* can be computed from a concept even without computing the entire lattice. However, computing the neighbors leads to an opportunity to compute the *entire lattice* and getting the lattice structure at the same time. We introduce all three algorithms in the following sections.

5.2.1 Derivation and concept

According to Definition 2, the derivation operator is defined over object sets as well as attribute sets. The result of the derivation depends on the given input set. Thus, we are speaking actually about two operations, one results in all common attributes of a given object set, and the other one results in the set of all objects which support a given attribute set. However, the same derivation symbol is used in [GW99] and [Lin00]. To avoid two separate functions for the derivation, we use an algebraic data type which provides one type constructor for object sets and an additional type constructor for attribute sets.

Listing 5.4: Data type for derivation (Concept)

```
61 data OASet o a = OSet (Set o) | ASet (Set a)
```

Then, it is possible to use the same data type for input and output of the derivation, whereas both can be either an object set or an attribute set. Now, we can define a function **prime** which maps a given **OASet** and a **Context** to the corresponding **OASet**. However, we define the function more similar to the mathematical notation. The derivation of a set A is usually denoted as A', confer Definition 2. The *formal context* is assumed to be given by the derivation context, i. e. the context in which the derivation occurs, and is, therefore, omitted. We support this notation by introducing a *state monad* which provides the *formal context* as a *state* for the derivation operator. Then, it is possible to define the function **prime**, the *first derivation*, without an explicit argument for a formal context.

Listing 5.5: Signature of prime (Concept)

```
prime :: (Ord o, Ord a, Ord v) =>
148   OASet o a ->
      State (Context o a v) (OASet o a)
```

Listing 5.6: Definition of prime (Concept)

```
231 prime (OSet os) = do
      (Context _ las li _) <- State.get
233   return (ASet (Set.fold (intersectS li) las os))
    prime (ASet as) = do
235   (Context los _ _ il) <- State.get
      return (OSet (Set.fold ((intersectS . (Map.map Map.keysSet)) il) los as))
```

The definition of the *second derivation* follows, then, straight from the first derivation. We choose a complete *formal concept* as result type, rather than a set of objects respective attributes, since the internal computation has always to compute the concept extend as well as the intent. This result type is not conform with the mathematical notation A'' for

a object respective attribute set A, but it provides still A'' in concept (A'', A'), if A is an object set and, correspondingly, (A', A''), if A is an attribute set. We even do not need to care about the derivation context for inner invocations of **prime**, if we define the *second derivation* function **second** also with the state monad to provide the derivation context. The derivation context is, then, automatically provided to all inner functions.

Listing 5.7: Signature of second (Concept)

```
second :: (Ord o, Ord a, Ord v) =>
158   OASet o a ->
      State (Context o a v) (Concept o a)
```

Listing 5.8: Definition of second (Concept)

```
240 second os@(OSet _) = do
      as@(ASet cas) <- prime os
242   (OSet cos)    <- prime as
      return (Concept cos cas)
244 second as@(ASet _) = do
      os@(OSet cos) <- prime as
246   (ASet cas)    <- prime os
      return (Concept cos cas)
```

We can use this inheritance property to design more complex functions and reduce them to **prime** or **second** without taking care of the derivation context. We show further applications in the next sections.

Using the introduced data types and functions, we can define a sample context and compute one of its concepts.

Listing 5.9: Example context (testBase)

```
test :: Context Int Char ()
6 test = fromOneValuedList [
```

```
  (1, "abc"),
8 (2, "bcd"),
  (3, "cde")]
```

To compute the result of **prime** or **second** on this context, we lift the context to a state, define a computation on this state, and request the result. This can be done by **evalState**.

Listing 5.10: Call of second on test with object set $\{2, 3\}$

```
Main> evalState (second (OSet (Set.fromList [2,3]))) test
2 ({2,3},{'c','d'})
```

5.2.2 Neighbors within a concept lattice

Lindig described in [Lin00] how upper neighbor concepts can be computed starting from a known concept and a given context. We adopt his algorithm to Haskell.

Listing 5.11: Neighbors algorithm by Lindig

```
NEIGHBORS ((A, B), (G, M, I))
2  min ← G \ A
   neighbors ← ∅
4  foreach g ∈ G \ A do
     B1 ← (G ∪ {g})'
6    A1 ← B1'
     if ((min ∩ (A1 \ A \ {g})) = ∅) then
8      neighbors ← neighbors ∪ {(A1, B1)}
     else
10     min ← min \ {g}
   return neighbors
```

The corresponding Haskell function **minNeighbors** takes a concept as argument and implicitly a context as state, in addition. The result is the set of concepts from the lattice

and the remaining content of **min**.

Listing 5.12: Signature of minNeighbors (Base.Neighbors)

```
minNeighbors :: (Ord o, Ord a, Ord v) =>
29  Concept o a ->
    State (Context o a v) (Set o, Set (Concept o a))
```

The loop from the original algorithm can be replaced by a folding, since it is iterating over elements in the set $G \setminus A$. In addition, we can reduce the condition which checks if $\min \cap (A_1 \setminus A \setminus \{g\})$ is empty within the loop. We use $\min \cap (A_1 \setminus \{g\})$, instead. The set min is defined as $G \setminus A$. Thus, it does not contain a single element of A and while looping it is only decreased in its elements. Then, $\min \cap (A_1 \setminus A \setminus \{g\}) = \min \cap (A_1 \setminus \{g\})$ holds within the loop, since

$$\begin{aligned} \min \cap (A_1 \setminus A \setminus \{g\}) &= (\min \cap A_1) \setminus \overbrace{(\min \cap A)}^{\varnothing} \setminus (\min \cap \{g\}) \\ &= (\min \cap A_1) \setminus (\min \cap \{g\}) \\ &= \min \cap (A_1 \setminus \{g\}) \end{aligned}$$

Listing 5.13: Definition of minNeighbors (Base.Neighbors)

```
minNeighbors (Concept cos cas) = do
46  los <- objectSet
    min <- return (los Set.\\ cos)
48  foldM nInnerLoop (min, Set.empty) (Set.toList min) where
      nInnerLoop (min, nbs) g = do
50      (Concept ncos ncas) <- second (OSet (Set.insert g cos))
        if Set.null (min `Set.intersection` (Set.delete g ncos))
52        then return (min, Set.insert (Concept ncos ncas) nbs)
          else return (Set.delete g min, nbs)
```

5.2.3 Complete lattice

Starting from the bottom concept, it is possible to compute an entire lattice by consequently computing upper neighbors. Moreover, the bottom concept, $(\varnothing'', \varnothing')$, is a well known node of each concept lattice. Other algorithms without the neighbor algorithm have been proposed. However, the approach by computing neighbors reveals the lattice structure, in addition to the set of concepts, as Lindig pointed out in [Lin00].

The function **lattice** takes the context as an implicit argument and results in a lattice.

Listing 5.14: Signature of lattice (Concept.Lattice)

```
lattice :: (Ord o, Ord a, Ord v) =>
  State (Context o a v) (Lattice o a)
```

Listing 5.15: Definition of lattice (Concept.Lattice)

```
lattice = do
  bottomConcept <- second (OSet Set.empty)
  let bottomNode = (bottomConcept, Set.empty, Set.empty)
  processNode (Map.singleton bottomConcept bottomNode) (Lattice Map.empty) where
    processNode nextConcepts cl
      | Map.null nextConcepts = return cl
      | otherwise             = do
        let (_, (concept, _, _)) = Map.findMin nextConcepts
        upNeighbors <- neighbors concept
        let neighborsCL = Set.fold (addNeighbor concept) cl upNeighbors
        let (updatedCL, nextcs) = insertConcept upNeighbors concept neighborsCL
        processNode nextcs updatedCL
    addNeighbor low curr (Lattice cl) = Lattice ncl where
      ncl = Map.insertWith mergeLow curr (curr, Set.empty, Set.singleton low) cl
      mergeLow (_, _, low) (curr, ups, lows) =
        (curr, ups, lows `Set.union` low)
```

```
    insertConcept currUps curr (Lattice cl) =
49    (Lattice ncl, nextcs) where
      ncl = Map.insertWith mergeUps curr (curr, currUps, Set.empty) cl
51    mergeUps (_, currUps, _) (curr, ups, lows) =
        (curr, ups `Set.union` currUps, lows)
53    (_, nextcs) = Map.split curr ncl
```

The function **lattice** invokes a tail recursive function **processNode** beginning with the bottom concept of the lattice. In **processNode**, neighbors of the actual concept are determined and inserted in the lattice. The recursion step invokes **processNode** with the next greater node in lattice according to to total order on concepts.

5.3 Time complexity

Lindig summarized already the time complexity for the basic operations. The worst-case time complexity of **second** with respect to a context (G, M, I) is $\mathbf{O}(|G| \times |M|)$. We can reason with our reference implementation, where **second** is reduced to two calls of **prime**, once to compute the concept extent and another time to compute the concept intent. The function **prime** can be reduced to **fold**, which folds **intersectS** into the incidence relation. The standard function **fold** on an arbitrary set A has time complexity $\mathbf{O}(|A| \times f)$ where f is the additional complexity of the folded function. In our implementation, this function **intersectS** performs a lookup operation on a binary tree and a set operation. The lookups are in $\mathbf{O}(\log |G|)$ respective $\mathbf{O}(\log |M|)$ and do, therefore, not add complexity to the all-in-all time. The complexity of the set intersection in **intersectS** is $\mathbf{O}(|G|)$ or $\mathbf{O}(|M|)$, respectively. Thus, the cardinality of G and M determine the time complexity for **fold** and, therefore, for **prime** to be $\mathbf{O}(|G| \times |M|)$. The function **second** calls **prime** twice, but this pre-factor does not lead to a higher complexity class.

Upper neighbor concepts can be computed in $\mathbf{O}(|G|^2 \times |M|)$, according to Lindig. We

reason with our function **minNeighbors** which can be reduced to a **fold** over **min**. The folded function calls **second** and computes several set operations over G. The set operations do not significantly increase the time complexity, since they are linear in $|G|$, whereas **second** provides already linear complexity in $|G|$. The set **min** consists of elements from G and operations on **min** are, therefore, also linear in $|G|$. Thus, the complexity can be determined by the folding operation in **min** which has at most $\mathbf{O}(|G|)$ multiplied by the complexity of **second**.

The complete lattice can be computed in $\mathbf{O}(|G|^2 \times |M| \times |L|)$. Unfortunately, the count of concepts $|L|$ in a lattice L does not functional depend on $|G|$ or $|M|$. We reason the time complexity of our **lattice** function. It can be reduced to **processNode**, which invokes itself recursively, whereas this recursion is limited by the count of concepts of the lattice. The reason is the condition to leave the recursion. If **nextConcepts** is an empty mapping, no further recursion step will happen. This mapping contains all concepts which are greater than the current concept. At the beginning of each invocation, **processNode** chooses the smallest concept of that mapping to be the current concept. This leads exactly to a recursion of $|L|$ steps, since we start with the bottom concept of the lattice. Within **processNode**, we invoke **neighbors**. Besides this invocation, there are only calls which have logarithmic or linear complexity with respect to $|G|$ or $|L|$ and introduce, therefore, neither a new parameter nor a greater complexity class. Thus, the time complexity of **lattice** can be determined by the complexity of **neighbors** and the size of the lattice $|L|$.

Chapter 6

Conclusions

We suggested an application of Formal Concept Analysis on privacy related contexts. It results in a concept for each linkability set. Contents of such a set are either pseudonyms or subjects. They are characterized by corresponding sets of all related data items. A set of pseudonyms or subjects, together with the corresponding characteristic set of data items, is a formal concept. One major achievement of this thesis is, therefore, the rearrangement of privacy related data in a conceptual manner.

We have, additionally, suggested linkability sets which contain messages instead of pseudonyms. Our investigation focused particularly on message contents as formal attributes of messages. They have been found very useful to determine the relation between linkable messages. It turned then out that message contents are very useful to determine even relations between subjects and messages. The union of two contexts, one following the definition of message contexts and the other one following the definition of subject contexts, leads to a concept lattice where also linkability between messages and subjects is considered.

The more data items belong to the attributes set in a formal context, the more fine-grained linkability sets belong to the corresponding concept lattice, thus, the better a linkability threat can be recognized. If we had access to the complete model of reality where

all pseudonyms are assigned to subjects, then our context definitions could even be used to compute concepts containing anonymity sets. A message which contains several data items can, then, be unambiguously assigned to one anonymity set by its set of characteristic data items. The structure of concept lattice can be used to determine data items which reduce anonymity or linkability sets least. They belong either to the attribute set of the same concept or to that one of a close lower neighbor.

Even complete (attacker) knowledge about a closed subject group is sufficient to identify anonymity sets reliably, if the data subject surely belongs to that group. The remaining part of the world does not need to be described in detail, then. Examples for such groups are all people within the same classroom or living in the same country.

However, decisions of a data subject can be supported even without a complete model of reality. Data items can be chosen as message contents in a way that makes pseudonyms not linkable to certain messages or other pseudonyms. Then, however, these other pseudonyms or messages have to be marked, before. Decisions can be made by checking for marked items, then.

One advantage of our approach is the preservation of connections between linkable items and these data items which caused the linkability. These connections are lost, if linkability is just expressed by probabilities. We see, therefore, an application of our approach in user-computer interaction where the reasons for linkability threads have to be explained to a data subject by arguing, for instance, with message contents in comparison to previously revealed data items.

In addition, the level of detail of formal contexts and lattices can be chosen according to the requirements of decision-making. In Chapter 3, we increased this level step by step. In case the formal context is changing frequently, it is preferable to reduce the level of detail as much as possible without loss of expressiveness, since this reduces also the complexity of computation significantly, as pointed out in Section 5.3. If a decision just depends on specific data items, then there is no need to include more than these data items in the

formal context. However, there is a point in including all available data items, in case the formal context is changed less often. The same conclusions can be drawn using the more complex lattice, since the structure of a less detailed lattice is still part of the complex lattice. Additional details can be represented as a nesting layers in a nested line diagram. If several decisions have to be taken using the same formal context, a highly detailed lattice can be stored and reused for each decision while recomputations are not necessary. The concept lattice needs even not to be computed entirely, if only parts of it are required to take the decisions. Methods have been presented to compute an arbitrary concept within the lattice and, basing on this, compute its neighbors.

A disadvantage of our approach is the lack of transitive propagation of linkability. It is yet not possible to point out a message which is linkable to another message which, again, is linkable to a specific subject. This can yet not be expressed by one concept whereas information theory approaches support this transitive propagation by adjusting probabilities accordingly.

Additionally, we have yet not considered a weighting of the privacy risk which arise from revealing data items. In fact, the risk of loosing privacy is not all the same for each data item. As we pointed out in Chapter 4, a credit card number is very good suited to track down the identity of a person, since the person itself tries to work against the card's abuse by someone else, and tries, therefore, to make it unambiguously assignable to itself. Interest in common food, for instance, is less critical with respect to re-identification of a person. In our approach, however, both data items appear the same way.

A disadvantage of all approaches, so far, is the static data lattice. It represents in a way background knowledge, but this knowledge can differ from one attacker to another. A company, for instance, would be able to link one of their customer numbers to a person whereas another company is usually not able to recognize any link, in this case. These different implications cannot be expressed in data lattices such as we suggest in Chapter 3 and even not with fixed probabilities.

Further research is needed on the combination of our conceptual approach and probabilistic approaches in order to use the advantages of both. In addition, an refinement of context definitions is necessary in order to support realistic use cases.

References

[ACC+05] Christer Andersson, Jan Camenisch, Stephen Crane, Simone Fischer-Hübner, Ronald Leenes, Siani Pearson, John Sören Pettersson, and Dieter Sommer, *Trust in prime*, Proceedings of the 5th IEEE Int. Symposium on Signal Processing and IT, December 2005, pp. 552–559.

[CFN90] D. Chaum, A. Fiat, and M. Naor, *Untraceable electronic cash*, CRYPTO '88: Proceedings on Advances in cryptology (New York, NY, USA), Springer-Verlag New York, Inc., 1990, pp. 319–327.

[CH02] Jan Camenisch and Els Van Herreweghen, *Design and implementation of the idemix anonymous credential system*, CCS '02: Proceedings of the 9th ACM conference on Computer and communications security (New York, NY, USA), ACM Press, 2002, pp. 21–30.

[DSCP02] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel, *Towards measuring anonymity*, Proceedings of Privacy Enhancing Technologies Workshop (PET 2002) (Roger Dingledine and Paul Syverson, eds.), Springer-Verlag, LNCS 2482, April 2002.

[FH01] Simone Fischer-Hübner, *It-security and privacy: Design and use of privacy-enhancing security mechanisms*, Lecture notes in computer science, vol. 1958, Springer-Verlag, 2001.

[FHAH05] Simone Fischer-Hübner, Christer Andersson, and Thijs Holleboom, *Framework v1*, Deliverable D14.1.a, PRIME – Privacy and Identity Management for Europe, Contract No. 507591, March 2005.

[GW99] Bernhard Ganter and Rudolf Wille, *Formal concept analysis: Mathematical foundations*, Springer-Verlag Berlin Heidelberg, 1999.

[HK05] Marit Hansen and Henry Krasemann, *Privacy and identity management for europe – prime white paper*, Deliverable D15.1.d, PRIME – Privacy and Identity Management for Europe, Contract No. 507591, July 2005.

[Lin00] Christian Lindig, *Fast concept analysis*, Work with Conceptual Structures – Contributions to ICCS 2000, Shaker Verlag, August 2000, pp. 152–161.

[Lin05] Tanja Linderborg, *Avidentifiera jobbansökningar – en metod för mångfald*, Tech. Report SOU 2005:115, Sveriges regering, December 2005.

[PC95] European Parliament and European Council, *Directive 95/46/ec on the protection of individuals with regard to the processing of personal data and on the free movement of such data*, Official Journal of the European Communities (1995), no. 281, 31–50.

[PH06] Andreas Pfitzmann and Marit Hansen, *Anonymity, unlinkablility, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology*, Technische Universität Dresden and ULD Kiel, v0.27 ed., February 2006.

[PW99] Susanne Prediger and Rudolf Wille, *The lattice of concept graphs of a relationally scaled context*, ICCS '99: Proceedings of the 7th International Conference on Conceptual Structures (London, UK), Springer-Verlag, 1999, pp. 401–414.

[SK03] Sandra Steinbrecher and Stefan Köpsell, *Modelling unlinkability*, Proceedings of Privacy Enhancing Technologies workshop (PET 2003) (Roger Dingledine, ed.), Springer-Verlag, LNCS 2760, March 2003.

[Tho99] Simon Thompson, *The craft of functional programming*, 2 ed., Addison-Wesley, 1999.

Appendix A

Reference implementation

Our reference implementations in Haskell have been built on the Hugs release of March 2005.

Listing A.1: Concept

```
module Concept (
  Attribute ( OneVal, ManyVal ),
  -- one type for prime and second
  OASet ( OSet, ASet ),
  -- formal context (G,M,I) (defined by incidence relation , I)
  Context,
  Incidence ( Incidence ),
  Scale,
  -- attribute -> plain attribute
  fromAttribute,
  -- incidence relation -> context
  fromIncidence,
  -- get object set from state
  objectSet,
  getObjectSet,
  -- get attribute set from state
```

```
    attributeSet,
18  getAttributeSet,
    -- get attribute value from state
20  attributeValue,
    getAttributeValue,
22  -- recover incidence relation from context
    incidence,
24  getIncidence,
    -- attribute list -> list of one-valued attributes
26  oneVals,
    -- list of (object, attribute set) -> one-valued context
28  fromOneValuedList,
    -- Concept
30  Concept ( Concept ),
    -- Lattice
32  Lattice ( Lattice ),
    -- first derivation: g' resp. m'
34  prime,
    getPrime,
36  -- second derivation: (g '', g') resp. (m',m'')
    second,
38  getSecond,
    -- context -> scale (you will prefer @mkScale@ in Concept.Lattice)
40  scaleFromContext,
    -- operations on scales
42  getScaleContext,
    getScaleLattice,
44  updateScaleLattice
    ) where
```

```
46
   -- MODULES --
48 import Prelude as Prelude
   import Data.Map as Map
50 import Data.Set as Set
   import Control.Monad.State as State
52
   -- TYPE DEFINITION --
54
   data Attribute a v = OneVal a | ManyVal a v
56   deriving (Ord, Eq)
   instance (Show a, Show v) => Show (Attribute a v) where
58   show (OneVal a)     = show a
     show (ManyVal a v) = "(" ++ show a ++ "@" ++ show v ++ ")"
60
   data OASet o a = OSet (Set o) | ASet (Set a)
62   deriving Show

64 data Context o a v =
     Context
66     (Set o)                    -- object set
       (Set a)                    -- attribute set
68     (Map o (Set a))            -- incidence relation
       (Map a (Map o (Maybe v))) -- value assignment (and reverse incidence)
70   deriving Show

72 data Incidence o a v = Incidence (Map o (Set (Attribute a v)))
     deriving Show
74
```

```
data Scale v a =
  Scale
    (Context v a ())            -- scale context
    (Maybe (Lattice v a))       -- scale lattice, if previously computed
  deriving Show

data Concept o a = Concept (Set o) (Set a)
  deriving Eq
instance (Ord o, Ord a) => Ord (Concept o a) where
  compare (Concept o1 _) (Concept o2 _)
    | o1Len > o2Len = GT
    | o1Len < o2Len = LT
    | otherwise     = compare o1 o2
    where
      o1Len = Set.size o1
      o2Len = Set.size o2
instance (Show o, Show a) => Show (Concept o a) where
  show (Concept os as) = show (os,as)

data Lattice o a = Lattice (Map (Concept o a) (
  Concept o a,                  -- concept
  Set (Concept o a),            -- upper neighbors
  Set (Concept o a)))           -- lower neighbors
instance (Show o, Show a) => Show (Lattice o a) where
  show (Lattice l) = show l

-- SIGNATURES --

-- flatten of possibly many-valued attributes
```

```
104 fromAttribute :: Attribute a v -> a

106 -- determination of context from a given incidence relation
    fromIncidence :: (Ord o, Ord a, Ord v) =>
108   Incidence o a v ->
      Context o a v
110
    -- access on the context state
112 objectSet :: (Ord o) =>
      State (Context o a v) (Set o)
114
    getObjectSet :: (Ord o) =>
116   Context o a v ->
      Set o
118
    attributeSet :: (Ord a) =>
120   State (Context o a v) (Set a)

122 getAttributeValue :: (Ord o, Ord a) =>
      a -> o ->
124   Context o a v ->
      Attribute a v
126
    attributeValue :: (Ord o, Ord a) =>
128   a -> o ->
      State (Context o a v) (Attribute a v)
130
    getIncidence :: (Ord o, Ord a, Ord v) =>
132   Context o a v ->
```

```
  Incidence o a v

incidence :: (Ord o, Ord a, Ord v) =>
  State (Context o a v) (Incidence o a v)

-- create one-valued attributes
oneVals :: (Ord a, Ord v) =>
  [a] -> Set (Attribute a v)

-- create one-valued contexts from lists
fromOneValuedList :: (Ord o, Ord a, Ord v) =>
  [(o,[a])] -> Context o a v

-- derrivation operator
prime :: (Ord o, Ord a, Ord v) =>
  OASet o a ->
  State (Context o a v) (OASet o a)

getPrime :: (Ord o, Ord a, Ord v) =>
  OASet o a ->
  Context o a v ->
  OASet o a

-- second derrivation
second :: (Ord o, Ord a, Ord v) =>
  OASet o a ->
  State (Context o a v) (Concept o a)

getSecond :: (Ord o, Ord a, Ord v) =>
```

```
162   OASet o a ->
      Context o a v ->
164   Concept o a

166 -- scale operations
    scaleFromContext :: (Ord a, Ord v) =>
168   Context v a () ->
      Scale v a
170
    getScaleContext :: (Ord a, Ord v) =>
172   Scale v a ->
      Context v a ()
174
    getScaleLattice :: (Ord a, Ord v) =>
176   Scale v a ->
      Lattice v a
178
    updateScaleLattice :: (Ord a, Ord v) =>
180   (Context v a () -> Lattice v a) ->
      Scale v a ->
182   Scale v a

184 -- IMPLEMENTATION --

186 fromAttribute (OneVal a)    = a
    fromAttribute (ManyVal a v) = a
188
    fromIncidence (Incidence ili ) = Context los las li il where
190   los = Map.keysSet ili
```

```
    las = (Set.unions . Map.elems) li
192  li  = Map.map (Set.map fromAttribute) ili
    il  = foldr revLi Map.empty (Map.toList ili)
194  revLi (o, as) fm = Set.fold insA fm as where
      insA (OneVal a)     = Map.insertWith Map.union a (Map.singleton o Nothing)
196   insA (ManyVal a v) = Map.insertWith Map.union a (Map.singleton o (Just v))

198 getObjectSet (Context os _ _ _) = os

200 objectSet = gets getObjectSet

202 getAttributeSet (Context _ as _ _) = as

204 attributeSet = gets getAttributeSet

206 getAttributeValue a o (Context _ _ _ il) = getAV oMap where
    oMap = Map.lookup a il
208  getAV Nothing = error "Concept.getAttributeValue:␣Attribute␣not␣present!"
    getAV (Just om) = res (Map.lookup o om)
210  res Nothing = error "Concept.getAttributeValue:␣Object␣not␣present!"
    res (Just Nothing) = OneVal a
212  res (Just (Just v)) = ManyVal a v

214 attributeValue a o = gets (getAttributeValue a o)

216 getIncidence ctx@(Context _ _ li _) = Incidence iRel where
    iRel = Map.mapWithKey assignValuesO li
218  assignValuesO o as = Set.map (assignValuesA o) as
    assignValuesA o a  = getAttributeValue a o ctx
```

```
220
   incidence = gets getIncidence
222
   oneVals = (Set.map OneVal) . Set.fromList
224
   fromOneValuedList =
226   fromIncidence . Incidence . (Map.map oneVals) . Map.fromList

228 intersectS rel oa oas =
      oas `Set.intersection` (Map.findWithDefault Set.empty oa rel)
230
   prime (OSet os) = do
232   (Context _ las li _) <- State.get
      return (ASet (Set.fold (intersectS li) las os))
234 prime (ASet as) = do
      (Context los _ _ il) <- State.get
236   return (OSet (Set.fold ((intersectS . (Map.map Map.keysSet)) il) los as))

238 getPrime = evalState . prime

240 second os@(OSet _) = do
      as@(ASet cas) <- prime os
242   (OSet cos)    <- prime as
      return (Concept cos cas)
244 second as@(ASet _) = do
      os@(OSet cos) <- prime as
246   (ASet cas)    <- prime os
      return (Concept cos cas)
248
```

```
getSecond = evalState . second

scaleFromContext context = Scale context Nothing

getScaleContext (Scale context _) = context

getScaleLattice (Scale _ Nothing) =
  error "Concept.getScaleLattice:␣Yet␣no␣lattice␣present!"

getScaleLattice (Scale _ (Just lattice)) = lattice

updateScaleLattice f (Scale context _) = Scale context lattice where
  lattice = Just (f context)
```

Listing A.2: Concept.Neighbors

```
module Concept.Neighbors (
  neighbors,
  getNeighbors,
  minNeighbors,
  latticeNeighbors
  ) where

-- MODULES --
import Prelude as Prelude
import Data.Map as Map
import Data.Set as Set
import Control.Monad.State as State
import Concept
```

```
15 -- TYPE DEFINITION --

17 -- SIGNATURES --

19 neighbors :: (Ord o, Ord a, Ord v) =>
     Concept o a ->
21   State (Context o a v) (Set (Concept o a))

23 getNeighbors :: (Ord o, Ord a, Ord v) =>
     Concept o a ->
25   Context o a v ->
     Set (Concept o a)
27
   minNeighbors :: (Ord o, Ord a, Ord v) =>
29   Concept o a ->
     State (Context o a v) (Set o, Set (Concept o a))
31
   latticeNeighbors :: (Ord o, Ord a) =>
33   Concept o a ->
     Lattice o a ->
35   Set (Concept o a)

37 -- IMPLEMENTATION --

39 neighbors concept = do
     (_, nbs) <- minNeighbors concept
41   return nbs

43 getNeighbors = evalState . neighbors
```

```
45 minNeighbors (Concept cos cas) = do
     los <- objectSet
47   min <- return (los Set.\\ cos)
     foldM nInnerLoop (min, Set.empty) (Set.toList min) where
49     nInnerLoop (min, nbs) g = do
         (Concept ncos ncas) <- second (OSet (Set.insert g cos))
51       if Set.null (min `Set.intersection` (Set.delete g ncos))
           then return (min, Set.insert (Concept ncos ncas) nbs)
53         else return (Set.delete g min, nbs)

55 latticeNeighbors (Concept cos cas) (Lattice cl) = Set.fromList neighborhood
     where
57     neighborhood = Prelude.foldr hideAncestors upperConcepts upperConcepts
       upperConcepts = Prelude.filter (greaterConceptTo cos) conceptList
59     conceptList = Prelude.map (\(concept, _, _) -> concept) (Map.elems cl)
       greaterConceptTo cos (Concept os _) = cos `Set.isProperSubsetOf` os
61     hideAncestors (Concept os _) cs =
         Prelude.filter (not . (greaterConceptTo os)) cs
```

Listing A.3: Concept.Lattice

```
   module Concept.Lattice (
2    lattice ,
     getLattice ,
4    mkScale
     ) where
6
   -- MODULES --
8  import Prelude as Prelude
```

```
import Data.Set as Set
import Data.Map as Map
import Control.Monad.State as State
import Concept
import Concept.Neighbors

-- TYPE DEFINITION --

-- SIGNATURES --

lattice :: (Ord o, Ord a, Ord v) =>
  State (Context o a v) (Lattice o a)

getLattice :: (Ord o, Ord a, Ord v) =>
  Context o a v ->
  Lattice o a

mkScale :: (Ord a, Ord v) =>
  Context a v () ->
  Scale a v

-- IMPLEMENTATION --

lattice = do
  bottomConcept <- second (OSet Set.empty)
  let bottomNode = (bottomConcept, Set.empty, Set.empty)
  processNode (Map.singleton bottomConcept bottomNode) (Lattice Map.empty) where
    processNode nextConcepts cl
      | Map.null nextConcepts = return cl
```

```
38      | otherwise              = do
          let (_, (concept, _, _)) = Map.findMin nextConcepts
40        upNeighbors <- neighbors concept
          let neighborsCL = Set.fold (addNeighbor concept) cl upNeighbors
42        let (updatedCL, nextcs) = insertConcept upNeighbors concept neighborsCL
          processNode nextcs updatedCL
44    addNeighbor low curr (Lattice cl) = Lattice ncl where
        ncl = Map.insertWith mergeLow curr (curr, Set.empty, Set.singleton low) cl
46      mergeLow (_, _, low) (curr, ups, lows) =
          (curr, ups, lows `Set.union` low)
48    insertConcept currUps curr (Lattice cl) =
        (Lattice ncl, nextcs) where
50      ncl = Map.insertWith mergeUps curr (curr, currUps, Set.empty) cl
        mergeUps (_, currUps, _) (curr, ups, lows) =
52        (curr, ups `Set.union` currUps, lows)
        (_, nextcs) = Map.split curr ncl
54

getLattice ct = evalState lattice ct
56

mkScale = (updateScaleLattice getLattice) . scaleFromContext
```

Listing A.4: Concept.Scaling

```
1  module Concept.Scaling (
     Scaling ( Scaling ),
3    scaleContext
     ) where
5

   -- MODULES --
7  import Prelude as Prelude
```

```
import Data.Set as Set
import Data.Map as Map
import Control.Monad.State as State
import Concept

-- TYPE DEFINITION --

data Scaling o1 a v o2 = Scaling (
  Map (Scale v a) (Set a) -> -- scales and affected  attributes
  Context o1 a v ->          -- many-valued context
  Context o2 a v)

-- SIGNATURES --

scaleContext :: (Ord o1, Ord a, Ord v, Ord o2) =>
  Scaling o1 a v o2 ->
  Map (Scale v a) (Set a) ->
  Context o1 a v ->
  Context o2 a v

-- IMPLEMENTATION --

scaleContext (Scaling f) = f
```

Listing A.5: Concept.Scaling.Plain

```
module Concept.Scaling.Plain (
  PlainScalable ( combine ),
  plainScaleContext,
  plainScaling
```

```
  ) where

6

  -- MODULES --
8 import Prelude as Prelude
  import Data.Set as Set
10 import Data.Map as Map
  import Control.Monad.State as State
12 import Concept
  import Concept.Scaling

14

  -- CLASS DEFINITION --

16

  class PlainScalable a where
18   combine :: a -> a -> a

20 -- SIGNATURES --

22 plainScale :: (Ord a, Ord v, PlainScalable a) =>
     Attribute a v ->      -- affected attribute
24   Scale v a ->          -- scale to use
     Set a

26

  plainScaleContext :: (Ord o, Ord a, Ord v, PlainScalable a) =>
28   Map (Scale v a) (Set a) -> -- scale and affected attributes
     Context o a v ->
30   Context o a v

32 plainScaling :: (Ord o, Ord a, Ord v, PlainScalable a) =>
     Scaling o a v o
```

```
34
   -- IMPLEMENTATION --
36
   plainScale (OneVal _) _ = error ("Concept.Scaling.Plain: " ++
38   "Attempt to scale one-valued attribute")
   plainScale (ManyVal a v) scale = Set.map (combine a) scaleAs where
40   (ASet scaleAs) = getPrime (OSet (Set.singleton v)) scaleContext
     scaleContext   = getScaleContext scale
42
   plainScaleContext = scaleContext plainScaling
44
   plainScaling = Scaling plain where
46   plain scales ctx = fromIncidence (Incidence scaledLi) where
       scaledLi = Map.foldWithKey processScale li scales
48     (Incidence li) = getIncidence ctx
       processScale scale as li = Set.fold processAttr li as where
50       processAttr attr li = Set.fold processObj li os where
           (OSet os) = getPrime (ASet (Set.singleton attr)) ctx
52         processObj obj li = replaceAttr mvAttr pAttrs obj li where
             pAttrs = plainScale mvAttr scale
54           mvAttr = Set.findMin (Set.filter findA setO)
             findA a = (fromAttribute a) == attr
56           setO = Map.findWithDefault Set.empty obj li

58 replaceAttr a as o li = Map.adjust repAttr o li where
     repAttr aSet = Set.union (Set.delete a aSet) (Set.map OneVal as)
```

Appendix B

Source code listings of formal contexts

Listing B.1: Table 3.2 on page 23

```
tabDataDriv :: Context String String ()
tabDataDriv = fromOneValuedList [
  ("d", ["d","n","b"]),
  ("i", ["i","n","b","N","a","h","c","p"]),
  ("n", ["n"]),
  ("b", ["b"]),
  ("N", ["N"]),
  ("a", ["a"]),
  ("h", ["h"]),
  ("c", ["c"]),
  ("p", ["p"])]
```

Listing B.2: Table 3.3 on page 24

```
tabDataVisit :: Context String String ()
tabDataVisit = fromOneValuedList [
  ("d", ["d","n","b"]),
  ("i", ["i","n","b","N","a","h","c","p"]),
```

```
    ("v", ["n","a"]),
    ("n", ["n"]),
    ("b", ["b"]),
    ("N", ["N"]),
    ("a", ["a"]),
    ("h", ["h"]),
    ("c", ["c"]),
    ("p", ["p"])]
```

Listing B.3: Table 3.4 on page 27

```
tabDataType :: Context String String ()
tabDataType = fromOneValuedList [
    ("d", ["d","n","b"]),
    ("i", ["i","n","b","N","a1","h","c","p","A"]),
    ("v", ["n","a2","A"]),
    ("n", ["n"]),
    ("b", ["b"]),
    ("N", ["N"]),
    ("a1", ["a1","A"]),
    ("a2", ["a2","A"]),
    ("h", ["h"]),
    ("c", ["c"]),
    ("p", ["p"])]
```

Listing B.4: Table 3.5 on page 28

```
tabMessMess1 :: Context String String ()
tabMessMess1 = fromOneValuedList [
    ("m1", ["n"]),
    ("m2", ["a"]),
```

```
61    ("m3", ["d"]),
      ("m4", ["v"])]
63
  tabMessScaled :: Context String String ()
65 tabMessScaled = fromOneValuedList [
      ("m1", ["n"]),
67    ("m2", ["a"]),
      ("m3", ["d","n","b"]),
69    ("m4", ["n","a"])]
```

Listing B.5: Table 3.6 on page 31

```
71 tabConnComm :: Context String String ()
   tabConnComm = fromOneValuedList [
73    ("m1", ["n","C2"]),
      ("m2", ["a","C1"]),
75    ("m3", ["d","n","b","C1"]),
      ("m4", ["n","a","C2"])]
```

Listing B.6: Table 3.8 on page 34

```
   tabConnPSubj :: Context String String ()
79 tabConnPSubj = fromOneValuedList [
      ("P1", ["S1","S4"]),
81    ("P2", ["S2"]),
      ("P3", ["S1","S3"]),
83    ("P4", ["S1","S3","S4"])]
```

Listing B.7: Table 3.10 on page 36

```
85 tabConnReverse :: Context String String ()
   tabConnReverse = fromOneValuedList [
```

```
87    ("S1", ["m1","m2","m3"]),
      ("S2", ["m4"]),
89    ("S3", ["m1","m3"]),
      ("S4", ["m1","m2"])]
```

Listing B.8: Table 3.13 on page 39

```
      tabConnOrigin1United :: Context String String ()
93    tabConnOrigin1United = fromOneValuedList [
      ("S1", ["d","n","b","a","R"]),
95    ("S2", ["n","a","R"]),
      ("S3", ["d","n","b","R"]),
97    ("S4", ["n","a","O"])]
```

Listing B.9: Scaled Table 4.2 on page 70

```
169   tabScenEBook :: Context String String ()
      tabScenEBook = fromOneValuedList tabScenEBookList
171   tabScenEBookList = [
      ("(mb01,P1)",["ib","O","C"]),
173   ("(mb01,B)",["ib","R"]),
      ("(mb02,P2)",["ib","ic","O","C"]),
175   ("(mb02,B)",["ib","ic","R"]),
      ("(mb03,P3)",["ib","ic","ir","O","C"]),
177   ("(mb03,B)",["ib","ic","ir","R"]),
      ("(mb04,P4)",["ib","ic","ir","ob","O","C"]),
179   ("(mb04,B)",["ib","ic","ir","ob","R"]),
      ("(mb05,P4)",["c","O","C"]),
181   ("(mb05,B)",["c","R"]),
      ("(mb06,P4)",["ib","ic","ir","ob","bv","R","C"]),
183   ("(mb06,B)",["ib","ic","ir","ob","bv","O"]),
```

```
    ("(P1,O)",["ib","O"]),
    ("(P1,C)",["ib","C"]),
    ("(P2,O)",["ib","ic","O"]),
    ("(P2,C)",["ib","ic","C"]),
    ("(P3,O)",["ib","ic","ir","O"]),
    ("(P3,C)",["ib","ic","ir","C"]),
    ("(P4,O)",["ib","ic","ir","ob","c","O"]),
    ("(P4,R)",["ib","ic","ir","ob","bv","R"]),
    ("(P4,C)",["ib","ic","ir","ob","c","bv","C"])]
```

Listing B.10: Scaled Table 4.4 on page 74

```
tabScenSoftware :: Context String String ()
tabScenSoftware = fromOneValuedList tabScenMarketList
tabScenSoftwareList = tabScenEBookList ++ [
    ("(ms01,P5)",["ic","is","O","C"]),
    ("(ms01,D1)",["ic","is","R"]),
    ("(ms02,P6)",["ic","is","O","C"]),
    ("(ms02,D2)",["ic","is","R"]),
    ("(ms03,P7)",["ic","is","O","C"]),
    ("(ms03,D3)",["ic","is","R"]),
    ("(ms04,P5)",["ic","is","os","O","C"]),
    ("(ms04,D1)",["ic","is","os","R"]),
    ("(ms05,P5)",["c1","O","C"]),
    ("(ms05,D1)",["c1","R"]),
    ("(ms06,P5)",["ic","is","os","sl","R","C"]),
    ("(ms06,D1)",["ic","is","os","sl","O"]),
    ("(P5,O)",["ic","is","os","c1","O"]),
    ("(P5,R)",["ic","is","os","sl","R"]),
    ("(P5,C)",["ic","is","os","c1","sl","C"]),
```

```
    ("(P6,O)",["ic","is","O"]),
213 ("(P6,C)",["ic","is","C"]),
    ("(P7,C)",["ic","is","C"]),
215 ("(P7,O)",["ic","is","O"]),
    ("(D1,O)",["ic","is","os","sl","O"]),
217 ("(D1,R)",["ic","is","os","c1","R"]),
    ("(D2,R)",["ic","is","R"]),
219 ("(D3,R)",["ic","is","R"])]
```

Listing B.11: Scaled Table 4.7 on page 78

```
221 tabScenMarket :: Context String String ()
    tabScenMarket = fromOneValuedList tabScenMarketList
223 tabScenMarketList = tabScenSoftwareList ++ [
    ("(mm001,P8)",["ie","O","C"]),
225 ("(mm001,M1)",["ie","R"]),
    ("(mm002,P8)",["ie","om01","O","C"]),
227 ("(mm002,M1)",["ie","om01","R"]),
    ("(mm003,P8)",["c","O","C"]),
229 ("(mm003,M1)",["c","R"]),
    ("(mm004,P8)",["a","O","C"]),
231 ("(mm004,M1)",["a","R"]),
    ("(mm005,P9)",["if","O","C"]),
233 ("(mm005,M1)",["if","R"]),
    ("(mm006,P9)",["if","om02","O","C"]),
235 ("(mm006,M1)",["if","om02","R"]),
    ("(mm007,P9)",["c","O","C"]),
237 ("(mm007,M1)",["c","R"]),
    ("(mm008,P9)",["a","O","C"]),
239 ("(mm008,M1)",["a","R"]),
```

```
    ("(mm009,P10)",["im","O","C"]),
    ("(mm009,M1)",["im","R"]),
    ("(mm010,P10)",["im","om03","O","C"]),
    ("(mm010,M1)",["im","om03","R"]),
    ("(mm011,P10)",["c","O","C"]),
    ("(mm011,M1)",["c","R"]),
    ("(mm012,P10)",["a","O","C"]),
    ("(mm012,M1)",["a","R"]),
    ("(P8,O)",["ie","om01","c","a","O"]),
    ("(P8,C)",["ie","om01","c","a","C"]),
    ("(P9,O)",["if","om02","c","a","O"]),
    ("(P9,C)",["if","om02","c","a","C"]),
    ("(P10,O)",["im","om03","c","a","O"]),
    ("(P10,C)",["im","om03","c","a","C"]),
    ("(M1,R)",["ie","om01","c","a","if","om02","im","om03","R"])]
```

Listing B.12: Scaled Table 4.9 on page 81

```
tabScenPaper :: Context String String ()
tabScenPaper = fromOneValuedList tabScenPaperList
tabScenPaperList = tabScenMarketList ++ [
    ("(mb11,P11)",["ib","ic","ir","obp","O","C"]),
    ("(mb11,B)",["ib","ic","ir","obp","R"]),
    ("(mb12,P11)",["ib","ic","ir","ob","bv","O","C"]),
    ("(mb12,B)",["ib","ic","ir","ob","bv","R"]),
    ("(mb13,P11)",["a","O","C"]),
    ("(mb13,B)",["a","R"]),
    ("(mb14,P11)",["c2","O","C"]),
    ("(mb14,B)",["c2","R"]),
    ("(mb15,P11)",["bb","R","C"]),
```

```
    ("(mb15,B)",["bb","O"]),
    ("(P11,O)",["ib","ic","ir","ob","obp","c2","a","bv","O"]),
    ("(P11,C)",["ib","ic","ir","ob","obp","c2","a","bv","bb","C"]),
    ("(P11,R)",["bb","R"]),
    ("(B,O)",["bb","O"]),
    ("(B,R)",["ib","ic","ir","ob","obp","c2","a","bv","R"])]
```

Listing B.13: Scaled Table 4.11 on page 83

```
tabScenMarketII :: Context String String ()
tabScenMarketII = fromOneValuedList tabScenMarketIIList
tabScenMarketIIList = tabScenPaperList ++ [
    ("(mm11,P12)",["ie","ix","im","om1","O","C"]),
    ("(mm11,M2)",["ie","ix","im","om1","R"]),
    ("(mm12,P12)",["c3","O","C"]),
    ("(mm12,M2)",["c3","R"]),
    ("(mm12,P12)",["a1","O","C"]),
    ("(mm12,M2)",["a1","R"]),
    ("(P12,O)",["ie","ix","im","c3","a1","om1","O"]),
    ("(P12,C)",["ie","ix","im","c3","a1","om1","C"]),
    ("(M2,R)",["ie","ix","im","c3","a1","om1","R"])]
```

Listing B.14: Scaled Table 4.13 on page 85

```
tabScenSupport :: Context String String ()
tabScenSupport = fromOneValuedList tabScenSupportList
tabScenSupportList = (Prelude.filter untouchedItems tabScenMarketIIList) ++ [
    ("(ms11,P13)",["ic","is","O","C"]),
    ("(ms11,X)",["ic","is","R"]),
    ("(ms12,P14)",["ic","is","ou","O","C"]),
    ("(ms12,X)",["ic","is","ou","R"]),
```

```
295   ("(ms13,P5)",["ic","is","ou","O","C"]),
      ("(ms13,X)",["ic","is","ou","R"]),
297   ("(ms14,P5)",["ic","is","os","sl","O","C"]),
      ("(ms14,X)",["ic","is","os","sl","R"]),
299   ("(P13,O)",["ic","is","O"]),
      ("(P13,C)",["ic","is","C"]),
301   ("(P14,O)",["ic","is","ou","O"]),
      ("(P14,C)",["ic","is","ou","C"]),
303   ("(P5,O)",["ic","is","os","ou","c1","sl","O"]),
      ("(P5,C)",["ic","is","os","ou","c1","sl","C"]),
305   ("(X,R)",["ic","is","os","ou","sl","R"])] where
      untouchedItems x
307     | x == ("(P5,O)",["ic","is","os","c1","O"])       = False
        | x == ("(P5,C)",["ic","is","os","c1","sl","C"]) = False
309     | otherwise                                        = True
```