Management of Uncertain Data

Towards unattended integration

Ander de Keijzer

CTIT Ph.D. Thesis Series No. 08-110
Centre for Telematics and Information Technology (CTIT)
P.O.Box 217 - 7500AE Enschede - The Netherlands

SIKS Dissertation Series No. 2008-04
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

ISBN: 978-90-365-2619-7
ISSN: 1381-3617 (CTIT Ph.D. Thesis Series no. 08-110)
Cover design: Eva de Keijzer
Printed by: PrintPartners Ipskamp, Enschede, The Netherlands

MANAGEMENT OF UNCERTAIN DATA
TOWARDS UNATTENDED INTEGRATION

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. W. H. M. Zijm,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 1 februari 2008 om 13.15 uur.

door

Ander de Keijzer

geboren op 2 december 1978
te Rotterdam

Dit proefschrift is goedgekeurd door:

Prof.dr. P.M.G. Apers (promotor)
Dr.ir. M. van Keulen (assistent-promotor)

Dankwoord

Na ruim vier jaar is het dan zover, er is een proefschrift. Het schrijven van een proefschrift is een individuele bezigheid waarmee wordt aangetoond dat je in staat bent zelfstandig onderzoek te verrichten. Als ik heel eerlijk ben, dan past dat niet zo goed bij me. Hoewel ik het erg leuk vind om onderzoek te doen en zelfstandig onderzoek doen in ieder geval zorgt dat er geen meningsverschillen zijn, vind ik samenwerken met anderen juist erg leuk. De discussies zijn dan misschien zelfs het leukst aan dat samenwerken.

Gelukkig heb ik de afgelopen paar jaar dan ook samengewerkt met heel wat mensen, binnen de leerstoel, maar zeker ook daarbuiten. Allereerst natuurlijk Maurice van Keulen, die me niet alleen gevraagd heeft om te solliciteren naar een plek bij de Database groep, maar met wie ik vervolgens ook binnen het MultimediaN project aan onzekerheid in databases heb gewerkt. De wekelijkse besprekingen, die soms niet zonder meningsverschillen waren, waren elke week een moment om naar uit te kijken. Peter Apers, die mij heeft aangenomen binnen de groep en hoewel hij niet vaak binnen de groep aanwezig was, tijdens besprekingen toch altijd precies wist waar mijn onderzoek over ging, maar belangrijker nog, de juiste vragen wist te stellen.

Tijdens de afgelopen jaren heb ik twee fijne kamergenoten gehad. De eerste, en langste, periode was dat Joeri. De wekelijkse quiz, het uitwisselen van recepten en de sfeer in de kamer hebben zeker bijgedragen aan het plezier waarmee ik naar mijn werk ging. Riham, mijn tweede kamergenoot, met wie ik zoveel mogelijk Nederlands heb geoefend en die inmiddels hopelijk gewend is aan mijn, soms enigszins gemene grapjes. Of course, I could have written this in English, but I am most confident that she can actually read the Dutch text as well.

Hoewel ik regelmatig moeite had om sprekers te vinden voor de Almost Weekend Meetings, of de Secret AIO Meetings, zoals ze ook wel genoemd worden, waren deze bijeenkomsten altijd een groot succes. Uiteraard is dat helemaal dankzij alle mede aio's van de DB groep. Ook de rest van de database groep heeft zeker bijgedragen aan de gezellige tijd, zowel tijdens pauzes, als gewoon tussendoor.

I am especially greatful to Jennifer Widom for having me at Stanford University for 6 months. The weekly Trio meetings, InfoLunches and also the personal discussions were both lively and educational. I am convinced that my visit to Stanford contributed tremendously not only to the thesis, but also to my way of working. I would also like to thank the other members of the Trio project and the whole InfoLab for making my time at Stanford very 'gezellig'.

Als AIO wordt er van je verwacht dat je onderzoek doet. Onderwijs, hoewel iedereen het leuk vindt als je hier een handje bij helpt, is niet een van de primaire taken. Het was echter het onderwijs geven dat me bij de Database groep bracht, aangezien ik Maurice ken omdat ik bij een van zijn vakken hielp bij het lesgeven. Ik ben ontzettend blij dat ik tijdens het promoveren de kans heb gekregen om niet alleen les te geven, maar zelfs mijn eigen vakken op te zetten. Heleen Miedema wil ik dan ook bedanken voor haar vertrouwen in mijn lesgeef capaciteiten. Hoewel ik formeel werk bij de Database groep, is TG toch altijd een tweede huis geweest voor mij. Ik wil dan ook Mieke Aitink, Marieke Hofman, Remke Burie, Benno Lansdorp en Astrid Dutrieux enorm bedanken voor de geweldige tijd.

Niet alleen was het altijd erg gezellig om even bij te praten met Ida, maar ook heeft zij ervoor gezorgd dat de organisatorische zaken rond het promoveren voor mij zo soepel mogelijk verliepen.

Mijn paranimfen, Eva en Margriet, ik ben blij en zeer vereerd dat jullie mij tijdens de promotie bij willen staan. Ik ken mezelf inmiddels, en de promotie zal voor mij een erg spannende dag zijn. Dat jullie daar samen met mij willen staan, maakt dat het allemaal wat makkelijker wordt.

En dan, als laatste, maar zeker niet de minste, mijn ouders, Coen en Joke. Hoewel jullie bijdrage aan het onderzoek en het proefschrift dan misschien niet direct te zien is, zou ik niet weten hoe ik het zonder jullie tot hier zou hebben gebracht.

Contents

Chapter 1

Introduction

Many of todays applications work with vast amounts of data. Take, for example, Sensor networks. These networks usually produce a steady stream of data for each of the sensors. The data from these sensors is stored. Next, this data is processed, where the data is usually aggregated and also this aggregated data is stored. One of the problems with sensor data is, that sensors are inherently uncertain. The data they produce can contain errors due to numerous causes. The reading from the sensor itself can be incorrect or the transmission may have introduced errors. The first is even almost a certainty, since most producers of sensors indicate the level of certainty of their sensors.

A database management system (DBMS) is responsible for storing data. However, the data stored in such a system needs to be correct at least according to the user of the DBMS at data insertion time. Although incorrect information can, of course, be stored in such a system, a user that later on retrieves data from the system will assume correctness of the data. In case of the sensor data, this can cause a problem, as most data will, to some extent, be incorrect, or at least imprecise.

In light of applications that use uncertain data, the DBMS should be able to *store, manage and query uncertain data.* The uncertainty associated with the data should be considered metadata, that is propagated whenever the user poses a query. This uncertainty can be stored in the form of *confidence scores.* Special operators should be available to allow the user direct access to these confidence scores, not only for querying, but also for manipulation of the scores.

Another application that can benefit from databases capable of storing uncertain data, are ambient database systems. Hardware becomes faster, cheaper and smaller on a daily basis. As a result Ambient database systems are becoming a reality. These database systems have a need to be as human

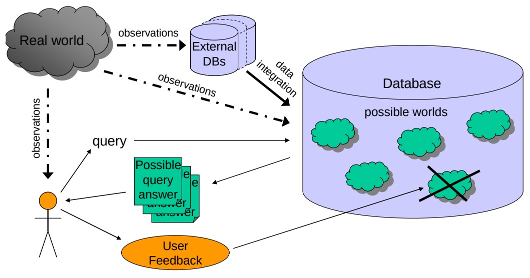


Figure 1.1: Information (Integration) Cycle

friendly as possible. Consider a PDA with telephone capabilities containing an address book application. All PDAs nowadays have synchronization capabilities, but also integration capabilities are supported. It would be infeasible to ask the owner of the PDA every time another PDA comes within range to manually check integration results. Instead, the address book application should be able to integrate the data itself and, if in doubt, store this uncertainty. At a later time, if the user wants to call somebody from the address book, all possible phone numbers are presented. If, at that time, the user discovers an error in the integration result, a feedback mechanism should be available to allow that particular possibility to be deleted.

In this thesis we use information integration as an application for uncertain data, much like the address book application presented. Information in an integration application evolves according to an *information cycle*. First, the data from several source documents is integrated into one integrated document. The integration approach taken in this thesis, is to postpone decisions on integration if there is uncertainty about equality of elements. This uncertainty introduces *possible states of the database*, called *possible worlds*. Next, a user of the integration application can query the integrated document. In an uncertain document, the query is posed in each of the possible worlds and the results from all worlds are grouped by object in the real world. The result of this query is presented to the user, and using a feedback technique introduced in Chapter 6, he can indicate if (part of) the result corresponds to the real world. This feedback then updates the data stored in the database according to the feedback statement. This information cycle shown in figure 1.1 shows how the possible world approach is used in the integration process and how feedback is processed. Using this information cycle, the integration process becomes unattended. This means that during the actual integration of data no human involvement is needed.

1.1 Information Integration

The area of information integration has been a topic of interest for many years. Numerous projects on the topic have been initiated, all focusing on different aspects, or approaching the problem from a different angle.

One of the challenges in information integration is finding correspondences between schemas of information sources. The last years, combining techniques, especially in integration of schemas has proven to be successful [Doa02]. After finding the correspondences in schemas, the actual data values have to be transformed to the new schema, and integrated into the new document. If two overlapping information sources are integrated, duplicate items will likely be present. These duplicates have to be eliminated. In order to accomplish this duplicate elimination, they first have to be found. Although this may sound like an easy task, this is not the case. The problem of finding these duplicates is known as *entity resolution*, *record linkage* or *data cleaning.*

1.1.1 Uncertain and Probabilistic Data

Earlier in this chapter we showed that there are many applications that deal with uncertain data in one way or another. There are many ways uncertainty can be dealt with. The way we use in this thesis is by specifying different possibilities for individual elements. The possibilities are mutually exclusive and are assigned a probability that indicates their likelihood of being the actual instance of that element. In this way, probability theory can be used to reason about possibilities, relations and queries.

1.2 Research questions

The previous sections illustrate that many applications benefit from allowing data to be uncertain. Integrating this uncertainty into a database management system and making it the responsibility of that system to maintain, propagate and manipulate the uncertainty, is the main research challenge in this work. A first research question that is addressed in this work, therefore is

Which additions to existing data models are necessary to be able to support uncertain data resulting from information integration?

In order to correctly work with the uncertainty associated with the data, the semantical foundation has to be defined. In addition, using this semantical foundation, the proposed model has to be complete and closed. Not only does the model need to be complete and closed, but also the results that are generated from queries and the queries themselves have to be intuitive.

The research question that is derived is

Which semantic foundation is needed to support intuitive querying on uncertain data?

In many research areas measures, testing frameworks and datasets are used to compare results from different projects with each other. Also comparing the results of one system from run to run, is possible when standardized measuring tools are available. We pose the following question to contribute to the solution of this problem

How can we measure uncertainty contained in documents and answer quality?

As an application for uncertain data, information integration seems promising. Especially, since it could potentially contribute to automating the process, or at least postponing user involvement. From the application side of uncertain data, we therefore have the following research question

How can uncertain database technology theoretically be applied in data integration?

If the user is no longer involved during the actual integration process, decisions on equality of elements is postponed. This results in large integrated documents. Therefore, the last research question we pose is

How can uncertain data be practically used in data integration?

1.3 Thesis structure

We start by giving an overview of the related work on both uncertain data and databases and one of its possible applications, information integration.

In Chapter 3 we introduce the probabilistic XML data model, enabling the system to capture probabilities associated with the data, mutual exclusiveness and dependencies. First we defined the semantics used in the probabilistic XML data model, which is that of possible worlds. We also introduce two new quality measures for uncertain data. The first measure, uncertainty density captures the amount of uncertainty in the document, without taking into account the probabilities associated with the data. The second measure, answer decisiveness does take these probabilities into account and indicates to what extent answers to queries are discriminative. In other words, how *easy* is it to choose between alternatives for elements based on the probabilities and the number of alternatives. These measures can be used to compare probabilistic documents and even systems. This chapter is largely based on work presented in [KKA05, dKvK07].

Chapter 4 deals with querying the probabilistic XML document, using the possible world semantics defined in Chapter 3. We show that some operations work across possible worlds instead of just combining the results from all possible worlds into one result. We also introduce new versions of precision and recall, adjusted to the setting of uncertain data. The traditional versions of precision and recall are not suitable for handling alternatives for documents. These new versions take into account the probability that is associated with a data item. Incorrect answers are only taking into account to the extent of their associated probability.

As one of the possible applications of uncertain data, information integration will be discussed in Chapter 5. First, we look at integration at schema level and then we use uncertain data to integrate at the data level. By storing uncertainty, the user of the integration process is no longer needed at integration time, but his involvement is postponed until query time.

The integration application produces documents that can become quite large. In Chapter 6 we introduce two methods to reduce the amount of uncertainty and with that, also the size of the resulting integrated document. The first method involves introducing world knowledge into the application. As a result, many of the possibilities in the resulting document become impossible. By keeping these knowledge rules as generic as possible, this method can be used across different domains. The second method introduced in this chapter is allowing the user of the system to give feedback on the results of a query. Any possible world contradicting the feedback statement is removed from the integrated document. Parts of this chapter are based on work presented in [KKA05].

In Chapter 7 we summarize and conclude this thesis. Also, we show future directions for research and provide some initial thoughts on these research questions.

Chapter 2

Related Research

2.1 Uncertain Data Models and Systems

In this section, we will visit existing projects and proposals for uncertain data. Different data models, such as relational and semistructured, as well as different uncertainty models, such as probabilistic and possibilistic are discussed. We also point to some other areas of interest in the uncertain data community.

2.1.1 Relational data

Several models for uncertain data have been proposed over the years. Initial efforts all focused on relational data [BGMP90] and also currently efforts are being made in the relational setting [LLRS97, BSHW06, BDM$^+$05, CSP05, AKO07a]. With relational data models, two methods to associate confidences with data are commonly used. The first method associates the confidence scores with individual attributes [BGMP90], whereas the second method associates these confidence scores with entire tuples [BSHW06].

Confidences associated with tuple level is also referred to as Type-1 uncertainty, whereas confidences associated with attribute level is referred to as Type-2 uncertainty. Type-1 and Type-2 uncertainty and a comparison between the two are further discussed in Chapter 3.

Table 2.1 shows examples of uncertain relational data using the two types of uncertainty. The first table uses attribute level uncertainty, whereas the second table uses tuple level uncertainty. Omitted confidence scores in the tables indicate a score of 1. Both tables contain address book information on persons named John and Amy and both capture uncertainty about their room and phone number. Table 2.1(a) uses Type-2 uncertainty and captures the fact that John either occupies room 3035 (with probability 40%), or

(a) Attribute level uncertainty

name	room	phone
John	3035 [.4]	1234
	3037 [.6]	
Amy	3122 [.6]	4321 [.6]
	3120 [.4]	5678 [.4]

(b) Tuple level uncertainty

name	room	phone	
John	3035	1234	.4
	3037		.6
Amy	3122	4321	.6
	3120	5678	.4

Table 2.1: Attribute and Tuple level uncertainty

3037 (with probability 60%), but certainly has phone number 1234. Amy, in this table, either occupies room 3122 (with probability 60%), or room 3120 (with probability 40%) and *independently* of the room has phone number 4321 (with probability 60%) or 5678 (with probability 40%). Table 2.1(b) uses Type-1 uncertainty and contains the same choices for room numbers and phone numbers for both persons, but in this case the room number and phone number for Amy are *dependent* on each other. If Amy occupies room 3122, then her phone number is 4321 analogously, if she occupies room 3120, then her room number is 5678. Observe that with tuple level uncertainty the expressiveness is larger, since dependencies between attributes can be expressed. This is impossible in the case of attribute level uncertainty. In the case of type-1 uncertainty it is, of course, possible to express the situation where both attributes are independent by enumerating all possibilities.

2.1.2 Semistructured data

Semistructured data, and in particular XML has also been used as a data model for uncertain data [HGS03, AS06]. As with the relational based models, there are two basic strategies. The first strategy is *event based* uncertainty, where choices for particular alternatives are based on specified events [AS06, HGS03]. The occurence of an event validates a certain part of the tree and invalidates the remainder of the tree. Using these events, possible worlds are created. Each combination for all events selects one of the possible worlds. In event based models, the events are independent of each other.

The other strategy for semistructured models is the *choice point based* uncertainty. With this strategy, at specific points in the tree a choice between the children has to be made. Choosing one child node, and as a result an entire subtree, invalidates the other child nodes. As with the event based strategy, possible worlds can be selected by choosing specific child nodes of choice points. The model presented in this thesis is based on the choice point strategy.

Figure 2.1.2 contains two XML documents containing identical information. The first document (Figure 2.1(a)) is a Fuzzy Tree according to [AS06], whereas the second tree (Figure 2.1(b)) is a probabilistic XML document according to the PXML model according of [HGS03]. Both XML documents are event based. Both documents contain address book information for a person named John. For this person, only a phone number, either 1234, or 4321 is stored. Figure 2.1(a) contains one event, called e. The name in the document is independent of the event and therefore, the name element is always present. In other words, the name element is associated with event *true* and therefore always present. If e is true, then the phone number is 1234, otherwise phone number is 4321. The likelihood of e being true is 30%. The same information captured in a choice point based model is presented in Figure 2.2. At each choice point, indicated by $\bigtriangledown$ one of the child elements can be chosen. The probability of each of the child nodes is given at the edge to that child node.

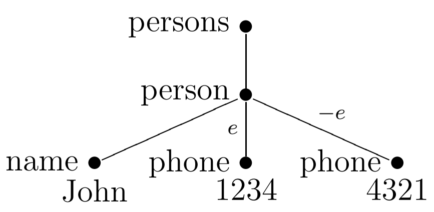
In Figure 2.1(b) the PXML of [HGS03], an event based model, is shown. In addition to the tree, the functions lch, $card$ and $\wp$ are provided. Function lch shows the child nodes of any given node o in the tree and associates a label l with the edge. Here, node S *has* a person node P. Function *card* gives the cardinality interval for each of the nodes in the tree, based on the labels of the edges. In this case, all cardinalities are exactly one. For node P this means that there is exactly one name edge, as well as exactly one phone edge. The final function $\wp$ provides probabilities for nodes that are uncertain. In this case, only T1 and T2 are uncertain. Since the cardinality constraint dictates that T1 and T2 are mutually exclusive, the probabilities for T1 and T2 add up to 1.

2.1.3 Confidence scores

The confidence scores associated with the data in uncertain databases, can be based on different paradigms. In many works, including this one, a probabilistic paradigm is used, but alternatively the possibilistic paradigm can be used, or even a more elaborate form such as cisets.

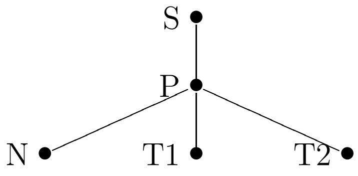
Probabilitic approach

With the probabilistic paradigm, all confidence scores are regarded as probabilities and are propagated as such. The result is, that at any given time, the total probability mass, or the sum of all probabilities, can't exceed 1. When calculating this probability mass, several things have to be taken into



event	**prob**
e	0.3

(a) Fuzzy Tree representation



o	l	$lch(o, l)$
S	person	{P}
P	name	{N}
P	phone	{T1, T2}

o	l	$card(o, l)$
S	person	[1, 1]
P	name	[1, 1]
P	phone	[1, 1]

$c \in PC(P)$	$\wp(P)(c)$
{T1}	0.3
{T2}	0.7

(b) PXML representation

Figure 2.1: Semistructured event based documents

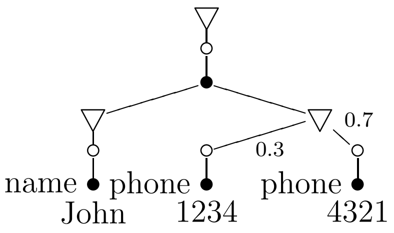


Figure 2.2: Semistructured choice point based document

account, such as local vs. global probabilities and dependencies. Type-1 probabilities, for example, are global probabilities when no joins are used. Type-2 probabilities on the other hand, are local to the tuple and only when alternatives for all of the attributes in a tuple are chosen, can the global, Type-1 probability can be calculated. Most data models and systems using probabilities assume independency among the tuples, but queries can create dependencies. If these dependencies are not taken into account, the calculated probability is incorrect. Systems using the probabilistic approach are MystiQ [BDM$^+$05] and Trio [Wid05]. Although the Trio system can be used with any other kind of paradigm, the standard paradigm used is probabilistic.

The data models and systems discussed so far all support discrete probabilistic distributions. Continuous distributions are another possibility for storing uncertainty about data. Here, the distribution itself also represents the data value of an attribute. Continuous uncertainty is supported by the ORION system [CSP05, CP05]. Consider, for example, a sensor application, that stores the data coming from a temperature sensor. Most producers of such sensors state that the sensor can report a temperature with a predefined uncertainty. We assume for this particular example that the actual temperature is normal distributed with the reported temperature as its mean and a static maximum deviation of $1°C$.

Possibilities

Instead of probabilistic, a possibilistic approach [BP04] can be taken for confidence scores. With possibility theory [Zad78] no assumptions have to be made about the dependency or independency of two uncertain attributes, tuples or elements. In addition, it is possible to express the possibility of an event occurring, without knowing its exact probability. In possibility theory, the *maximum* confidence score of any data element can't exceed 1, but there is no upper bound on the sum of the confidence scores.

Cisets

A third method to model uncertainty about data, is by means of confidence index sets, or cisets [Nai03]. A ciset is a pair $< \alpha, \beta >$ with $\alpha, \beta \in [0, 1]$. A ciset can be thought of as a mapping $F : S \to C$, where S is a set and F assigns to each element $x \in S$ two degrees of confidence α and β. The degree α is a confidence value that specifies the confidence of $x \in S^c$, with S^c the complement of S. Confidence value β specifies the confidence of $x \in S$. For each $< \alpha, \beta >$ holds that $0 \leq \alpha + \beta \leq 2$. This indicates the evidence that a certain $x \in S$ is not used to support $x \in S^c$. One observation that can be made, is that if the ciset space is restricted according to $\alpha = 1 - \beta$, cisets are reduced to the probabilistic paradigm. In this way, we can regard cisets to be a generalization of probability theory. With cisets, we can simultaneously store evidence in favor of and contradictory with an element, or event.

2.1.4 Prominent Projects

There are some prominent projects in the area of uncertain databases, some of which we already referred to earlier. Here, we give an overview of these projects, including their characteristics. Currently probably the best known, although certainly not the first project on uncertain data is Trio [Wid05, MTdK+07, BSHW06]. In this project at Stanford University, a database is developed supporting both uncertainty and lineage[1] as first class citizens in the system. Trio uses a relational data model, with a probabilistic approach. Although, as mentioned earlier, the user of Trio is free in *plugging in* its own arithmatic for confidence computation.

A project more focused on the complexity, efficiency and optimization of querying uncertain data is MYSTIQ at the University of Washington [BDM+05, DS07, RDS06]. This project is also based on the relational model and uses a probabilistic approach.

An earlier project from the University of Maryland is ProbVIEW [LLRS97]. ProbView, as Trio and MYSTIQ, is a relational database and uses a probabilistic approach. From the same university, a couple of years later, came PXML [HGS03], an XML based probabilistic database.

The last project we mention here, is MayBMS at Cornell University [AKO07b]. MayBMS is a relational system that uses a finite world-set decomposition. The confidence computation in MayBMS is based on probability theory.

2.2 Data Inconsistency

Repairing data inconsistencies can also be regarded as creating uncertainty in the data [Wij07]. Consider for example Table 2.2. According to the schema the name is a key, indicated by underlining the attribute. Since name is a key, it cannot be replicated in the table. We immediately see a violation of this constraint, because there are two instances where the name is “John”. A possible solution to this problem is creating two mutually exclusive *possible worlds*. One where (John, 3035, 1234) exists, and one where (John, 3037, 8765) exists.

[1]Lineage indicates *where the data came from*

<u>**name**</u>	**room**	**phone**
John	3035	1234
John	3037	8765
Amy	3122	4321
Anna	3120	5678

Table 2.2: Inconsistent Data

2.3 Querying Uncertain Data

In databases supporting uncertainty that are based on the relational model, SQL is the standard query language. Of course, SQL is extended with support for querying, updating and manipulating probabilities. Also, regular SQL expressions are rewritten to cope with probabilities associated with the data [BSHW06, MTdK$^+$07, AKO07b, RDS06, DS96]. In Trio, for example, the system rewrites a TriQL[2] statement to a regular SQL statement. All uncertainty associated with the data is stored in an underlying regular relational database. In case of Trio, even the lineage is stored in the same relational database. The Trio interface *hides* this Trio metadata from the user of the Trio system. Semantics for querying nowadays is most commonly the possible world semantics. In Chapter 4 we will elaborate on semantics and querying.

2.4 Complexity and Optimization

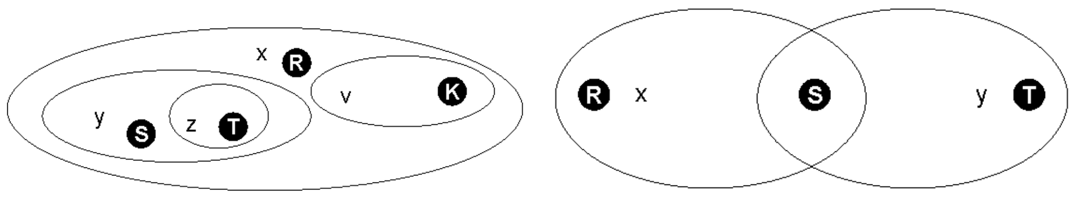
Directly related to querying of uncertain data, is studying the time complexity and optimizing queries for uncertain data. The complexity problem in querying uncertain data does not come from the data itself, but from the confidence computation that is needed to calculate confidences on query results. Currently, most work in the area of complexity analysis is being done in the MystiQ project [BDM$^+$05, RDS06, DS07]. For purely hierarchical queries like[3]

```
Q(w) :- R(x), S(x,w,y), T(x,y,z),K(x,v,w)
```

In this example Q is the query with parameter w. R, S, T and K are *subqueries* and v, x, y, z are parameters.

[2]Trio Query Language

[3]In DataLOG notation. Example taken from [Suc, Suc06].



(a) Hierarchical Query (b) Non hierarchical Query

Figure 2.3: Hierarchical and non-hierarchical query

The complexity of the query is PTIME, but as soon as queries are non hierarchical, it becomes $\#P$-complete [RDS06]. As is, for example, the following query[4]:

```
Q :- R(x), S(x, y), T(y)
```

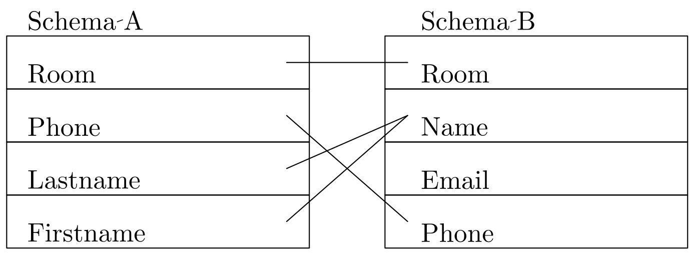
The fact that the first query is hierarchical and the second isn't, can be easily observed in Figure 2.3. Ellipses indicate parameters in the query, the name of the parameter is shown in the ellipse. Names of subqueries are shown in black circles. As long as ellipses don't partially overlap, but merely subsume another ellipse, the query is hierarchical. This is, because if ellipses partially overlap subqueries don't use a subset of each others parameters.

Currently, the Trio solution to the problem of expensive confidence computation is to postpone confidence computation as much as possible until the user needs the confidences. Since the lineage of the data is stored in Trio, confidence computation can be postponed until needed.

2.5 Information Integration

The amount of work on information integration is enormous. The topic has been studied for several decades already and will remain a research question for many more to come. This is due to the semantics captured by the schema. This semantics is impossible to determine by a machine and human involvement will always be necesarry to make the final decision about equality of schema elements. The first challenge in integration, is matching the elements from one data source onto the element from another data source. The result of this process is a mapping between the two documents relating not only the elements, but also providing mapping functions that indicate *how* the data is transformed from one document to the other. See Figure 2.4 for a schematic illustration.

[4]Example taken from [Suc, Suc06]



B.Room	=	A.Room
B.Name	=	A.Firstname + <space> + A.Lastname
B.Phone	=	A.Phone
B.Email	=	NULL

Figure 2.4: Schematic representation of mapping result

A recent overview of integration focused more on schema integration is given in [DH05]. The Learning Source Descriptions (LSD) project [DDH01, Doa02, DDH03] from the same authors is widely recognized as a big step forward in the schema integration field. In this project, base learners, are trained for specific parts of semantical domains. A meta learner is trained on the results of the base learners. As a result, the meta learner can combine the results of the base learners, based on the specific schemas that are being integrated.

One of the main problems in data integration, is finding mappings between elements from two (or more) schemas. Different projects in both relational and semistructured settings have been initiated [Smi06, CGMH$^+$94, GMPQ$^+$97, dV06, Bos07, Vis07]. Figure 2.5 gives an overview of the architecture of a typical integration system [Smi06, dV06]. In all of these projects mentioned, different techniques are used to find matches, or mappings between elements. Approaches from AI, clustering, semi automated methods are some of the techniques applied in these projects. Finding mappings is the task of component 2 in figure 2.5.

Although, the schema integration phase is an important and difficult part of the entire integration process, it is not the main focus of this thesis. The focus of this thesis is the part that takes place after schema matching. Usually when two information sources are integrated, there is an overlap in data instances. When data instances have to be integrated, decisions on equality of those instances have to be made. In this thesis we present a method to make this process unattended, i.e. the user does not need to be actively involved during this process.

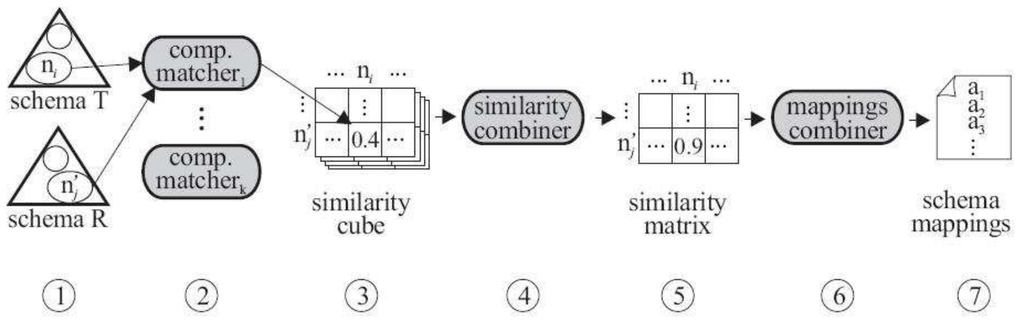


Figure 2.5: Architecture of a schema integration system

Chapter 3

Modeling Uncertain Data

In this chapter we will discuss the data model for probabilistic or uncertain XML. A formal definition of the uncertain XML structure is given and the semantics behind the data model is discussed. Some properties of the model are highlighted and two storage improvements on the data model are presented.

3.1 Possible Worlds

The semantics used in the probabilistic XML model is that of the possible worlds. This semantics is used in several other uncertain and probabilistic models and projects and is an intuitive interpretation of the uncertainty associated with the data.

If a database is considered to hold information on *real world objects*, then an uncertain database holds possible representations of those real world objects. Each of those possible representations can have an associated probability. If one of the possibilities for a real world object is *not to exist*, then this also is considered to be one of the possible representations.

A possible world is constructed by choosing one representation for each of the real world objects in the database. Instead of one database, an uncertain database can be seen as a set of possible databases. Or, if a database represents (part of) the real world, an uncertain database represents a set of (parts of) possible worlds. As an example consider Table 3.1. In this table information about 2 people, named John and James, is stored. For both "John" and "James" the phone number is uncertain and in both cases there are two possibilities, or alternatives for the value of the attribute Phone. From this table $2 \times 2 = 4$ possible worlds can be constructed, all combinations between different possibilities for each of the people stored in the database.

(a) Source Database

Addresses

Name	**Phone**
John	555-1234
John	555-4321
James	555-5678
James	555-8765

(b) Possible Worlds

World 1

Name	**Phone**
John	555-1234
James	555-5678

World 2

Name	**Phone**
John	555-1234
James	555-8765

World 3

Name	**Phone**
John	555-4321
James	555-5678

World 4

Name	**Phone**
John	555-4321
James	555-8765

Table 3.1: Construction of Possible Worlds

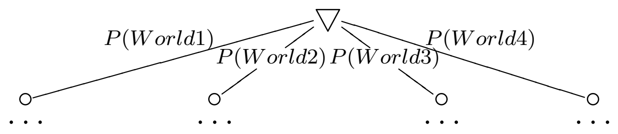


Figure 3.1: Possible world representation of Address Book Example (XML)

3.2 Probabilistic XML

In this section, we will introduce the notion of probabilistic XML, using the possible world approach described earlier. Following the possible world approach, we store possible appearances of the database instead of one actual appearance using XML as underlying data model. Consequently, our data model is a probabilistic XML data model. The simplest way to construct uncertain XML using the possible world approach, is by enumerating all possible worlds in different subtrees and combining those subtrees into one XML document. If desired, probabilities indicating the relative likelihood of each of the worlds, can be associated with the subtrees. This representation is called the *possible world representation.* Figure 3.1 shows the probabilistic XML representation of the possible worlds in Table 3.1. In this figure the actual XML nodes are replaced by $(\cdots)$ to increase readability. These should be replaced by certain XML trees representing that particular world.

Figure 3.1 shows that only the top level of the document contains a *choice* and all of the subtrees of the top level nodes are *certain XML documents.*

Since most possible worlds largely overlap, most nodes in the document are duplicated in several possible worlds. Therefore, the possible world representation, although theoretically interesting, semantically sound and easy to understand, is not practical. However, it is used to demonstrate concepts and functionality in the probabilistic XML DBMS. The possible world representation is used as a starting point and in subsequent sections we will show improvements on this general possible world representation.

3.3 Compact Representation

This section builds upon normal XML and the possible world model described earlier. We improve the storage model by reducing redundancy in storage. Our model is viewed as a tree, made up of nodes, containing subtrees. We distinguish between three different kinds of nodes to be able to store possibilities and associated probabilities. The use of three different kinds of nodes increases expressiveness, as we will later show.

Since order is important in XML, we first introduce some notation for handling sequences.

Notational convention 1 *Analogous to the powerset notation* $\mathbb{P}$*, we use a* power sequence *notation* $\mathbb{S}\,A$ *to denote the domain of all possible sequences built up of elements of* A*. We use the notation* $[a_1, \ldots, a_n]$ *for a sequence of* n *elements* $a_i \in A$ $(i = 1..n)$*. We use set operations for sequences, such as* $\cup, \exists, \in$*, whenever definitions remain unambiguous.*

We start by defining the notions of *tree* and *subtree* as abstractions of an XML document and fragment. We model a tree as a node and a sequence of child subtrees.

Definition 2 *Let* $n = \{id, tag, kind, attr, value\}$ *be a node, with*

- *id the node identity*
- *tag the tag name of the name*
- *kind the node kind*
- *attr the list of attributes, which can be empty*
- *value the text value of the node, which can be empty*

Equality on nodes is defined as equality on all of their properties. Deep-equality on nodes is defined as equality on nodes and their subtrees. We indicate that a certain node n is a *root node* by $\overline{n}$. Except for equality, however, we abstract from the details of nodes.

Definition 3 *Let $\mathcal{N}$ be the set of* nodes*. Let $\mathcal{T}_i$ be the set of* trees *with maximum level i inductively defined as follows:*

$$\begin{aligned}
\mathcal{T}_0 &= \{(n, \emptyset) \mid n \in \mathcal{N}\} \\
\mathcal{T}_{i+1} &= \mathcal{T}_i \cup \{(n, ST) \mid n \in \mathcal{N} \\
&\qquad \wedge ST \in \mathbb{S}\mathcal{T}_i \\
&\qquad \wedge (\forall T \in ST \bullet n \notin \mathcal{N}^T) \\
&\qquad \wedge (\forall T, T' \in ST \bullet T \neq T' \\
&\qquad\quad \Rightarrow \mathcal{N}^T \cap \mathcal{N}^{T'} = \emptyset)\}
\end{aligned}$$

where $\mathcal{N}^T = \{n\} \cup \bigcup_{T' \in ST} \mathcal{N}^{T'}$. Let $\mathcal{T}_{\text{fin}}$ be the set of finite trees*, i.e., $T \in \mathcal{T}_{\text{fin}} \Leftrightarrow \exists i \in \mathbb{N} \bullet T \in \mathcal{T}_i$. In the sequel, we only work with finite trees.*

Definition 3 requires the document to be a tree instead of a graph. A node has a sequence of child nodes, which can be empty and can have only one parent.

Since we will often work with entire subtrees instead of single nodes, we define some functions to obtain a subtree. We obtain a subtree from a tree T by indicating a node n in T which is the root node of the desired subtree. We also define a function child that returns the child nodes of a given node in a tree.

Definition 4 *Let* $\mathsf{subtree}(T, n)$ *be the* subtree *within $T = (\overline{n}, ST)$ rooted at n.*

- $\mathsf{subtree}(T, n) = \begin{cases} T & \text{if } \overline{n} = n \\ \mathsf{subtree}(T', n) & \text{otherwise} \end{cases}$

where T' such that $(n', T') \in ST \wedge n \in \mathcal{N}^{T'}$.
For $\mathsf{subtree}(T, n) = (n, [(n_1, ST_1), \ldots, (n_m, ST_m)])$,
let $\mathsf{child}(T, n) = [n_1, \ldots, n_m]$.

3.3.1 Probabilistic Tree

The central notion in our model is the *probabilistic tree.* In an ordinary XML document, all information is certain. In probabilistic XML each XML node can have zero or more possibilities, or alternatives. More generally, if we consider a node to be the root node of a subtree, then there may exist zero or more possibilities for an entire subtree. We model a probabilistic tree by introducing two special kinds of nodes:

1. probability nodes depicted as $\bigtriangledown$, and

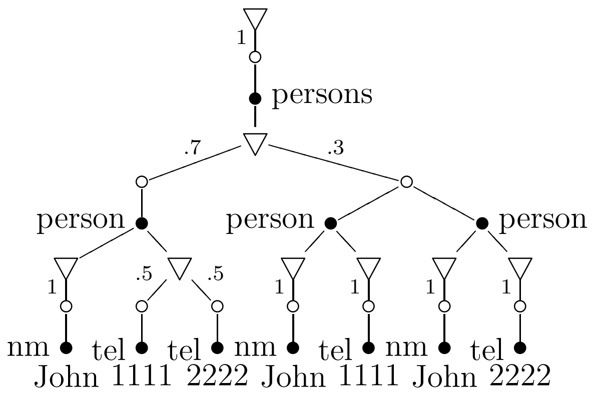


Figure 3.2: Example probabilistic XML tree.

2. possibility nodes depicted as ∘, which have an associated probability.

The root of a probabilistic XML document is always a probability node. Children of a probability node are always possibility nodes and enumerate all possibilities. The probabilities associated with the possibility nodes sum up to at most 1, or all probabilities of sibling possibility nodes are *unknown*.

Ordinary XML nodes are depicted as • and are always child nodes of possibility nodes. A probabilistic tree is well-structured, if the children of a probability node are possibility nodes, the children of a possibility node are XML nodes, and the children of XML nodes are probability nodes. Using this layered structure, each level of the tree only contains one kind of nodes.

Figure 3.2 shows an example of a probabilistic XML tree. The tree represents an XML document with a root node 'persons' (which exists with certainty). The root node has either one or two child nodes 'person' (with probabilities .7 and .3, respectively). In the case there is only one child, the name of the person is 'John' and the telephone number is either '1111' or '2222'. The probabilities for both phone numbers are uniformly distributed. The second case, where there are two persons with name 'John' is less likely if we consider *names* to be a key like element. However, we can store this more unlikely situation and in that case, the information of both persons is certain, i.e., they both have name 'John' and one has telephone number '1111' and the other has phone number '2222'.

In Chapter 5 we will use information integration as an application of probabilistic XML. Figure 3.2 can be seen as a possible result of two documents having been integrated. One document stating the telephone number of a person named 'John' to be '1111', and the other stating the telephone number of a person named 'John' to be '2222'. It is uncertain if both represent the same person (in the real world). A data integration matching rule apparently determined that, with a probability of .7, they represent the same person. Therefore, the combined knowledge of the real world is described accurately by the given tree.

A probabilistic tree is defined as a tree, a kind function that assigns node kinds to specific nodes in the tree, and a prob function that assigns probabilities to possibility nodes. The root node is defined to always be a probability node. A special type of probabilistic tree is a *certain* one, which means that all information in it is certain, i.e., all probability nodes have exactly one possibility node with an associated probability of 1.

Definition 5 *A* probabilistic tree $PT = (T, \mathsf{kind}, \mathsf{prob})$ *is defined as follows*

- $\mathsf{kind} \in (\mathcal{N} \to \{\underline{\mathsf{prob}}, \underline{\mathsf{poss}}, \underline{\mathsf{xml}}\})$
- $\mathcal{N}^T_k = \{n \in \mathcal{N}^T \mid \mathsf{kind}(n) = k\}$.
- $\mathsf{kind}(\overline{n}) = \underline{\mathsf{prob}}$ *where* $T = (\overline{n}, ST)$
- $\forall n \in \mathcal{N}^T_{\underline{\mathsf{prob}}} \forall n' \in \mathsf{child}(T, n) \bullet n' \in \mathcal{N}^T_{\underline{\mathsf{poss}}}$
- $\forall n \in \mathcal{N}^T_{\underline{\mathsf{poss}}} \forall n' \in \mathsf{child}(T, n) \bullet n' \in \mathcal{N}^T_{\underline{\mathsf{xml}}}$
- $\forall n \in \mathcal{N}^T_{\underline{\mathsf{xml}}} \forall n' \in \mathsf{child}(T, n) \bullet n' \in \mathcal{N}^T_{\underline{\mathsf{prob}}}$
- $\mathsf{prob} \in \mathcal{N}^T_{\underline{\mathsf{poss}}} \rightarrowtail [0, 1]$
- $\forall n \in \mathcal{N}^T_{\underline{\mathsf{prob}}} \bullet ((\sum_{n' \in \mathsf{child}(T,n)} \mathsf{prob}(n')) = 1 \vee (\forall n' \in \mathsf{child}(T, n) \bullet \mathsf{prob}(n') = \bot))$.

Where $A \rightarrowtail B$ *is a partial function.*

A probabilistic tree $PT = (T, \mathsf{kind}, \mathsf{prob})$ *is* certain *iff there is only one possibility node for each probability node, i.e.,* $\mathsf{certain}(PT) \Leftrightarrow \forall n \in \mathcal{N}^T_{\underline{\mathsf{prob}}} \bullet |\mathsf{child}(T, n)| = 1$. *To clarify definitions, we use* b *to denote a probability node,* s *to denote a possibility node, and* x *to denote an XML node.*

Subtrees under probability nodes denote *local* possibilities. In the one-person case of Figure 3.2, there are two local possibilities for the phone number, it is either '1111' or '2222'. The other uncertainty in the tree are the possibilities that there are one or two persons. Viewed globally and from the perspective of a device with this data in its database, the real world could look like one of the following

- one person with name 'John' and phone number '1111' (probability $.5 \times .7 = .35$),
- one person with name 'John' and phone number '2222' (probability $.5 \times .7 = .35$), or
- two persons with name 'John' and respective phone numbers '1111' and '2222' (probability .3).

We get these possible worlds by making a decision for one of the possibility nodes at each of the probability nodes. For this reason, we also refer to probability nodes as *decision points*.

Definition 6 *A certain probabilistic tree PT' is a* possible world *of another probabilistic tree PT, i.e., $\mathsf{pw}(PT', PT)$, with* probability $\mathsf{pwprob}(PT', PT)$ *iff*

- $PT = (T, \mathsf{kind}, \mathsf{prob}) \wedge PT' = (T', \mathsf{kind}', \mathsf{prob}')$
- $T = (\overline{n}, ST_{\overline{n}}) \wedge T' = (\overline{n}, ST'_{\overline{n}})$
- $\exists s \in \mathsf{child}(T, \overline{n}) \bullet \mathsf{child}(T', \overline{n}) = [s]$
- $X = \mathsf{child}(T, s) = \mathsf{child}(T', s)$
- $\forall x \in X \bullet \mathsf{child}(T, x) = \mathsf{child}(T', x)$
- $B = \bigcup_{x \in X} \mathsf{child}(T, x)$
- $\forall b \in B \bullet PT_b = \mathsf{subtree}(PT, b)$
  $\wedge PT'_b = \mathsf{subtree}(PT', b)$
  $\wedge \mathsf{pw}(PT'_b, PT_b)$
- $\forall b \in B \bullet p_b = \mathsf{pwprob}(PT_b, PT'_b)$
- $\mathsf{pwprob}(PT', PT) = \mathsf{prob}(s) \times \prod_{b \in B} p_b$

The set of all possible worlds of a probabilistic tree PT is
$\mathsf{PWS}_{PT} = \{PT' \mid \mathsf{pw}(PT', PT)\}$.

A probabilistic tree is a compact representation of the set of all possible worlds, but there is not necessarily one unique representation. The optimal representation is the one with the least number of nodes obtained through a process called simplification.

Definition 7 *Two probabilistic trees PT_1 and PT_2 are* equivalent *iff* $\mathsf{PWS}_{PT_1} = \mathsf{PWS}_{PT_2}$. *$PT_1$ is* more compact *than PT_2 if* $\left|\mathcal{N}^{PT_1}\right| < \left|\mathcal{N}^{PT_2}\right|$. *The transformation of a probabilistic tree to an equivalent more compact one is called* simplification.

The number of possible worlds captured by a probabilistic tree is determined by the number of decision points and possibilities at those points. We also define a function leaf that returns all the leaf nodes of a tree.

The number of possible worlds defined by the tree PT, $N_{PT}^{PW(T)}$ is equal to the number of possible worlds at node $\overline{n}$, defined by $N_{\overline{n}}^{PW(T)}$ where

- $\mathsf{leaf}(T) = \{n | n \in \mathcal{N}^T \bullet \mathsf{child}(n) = \emptyset\}$
- $N_n^{PW(T)} = 1$, if $n \in \mathsf{leaf}(T)$
- $N_n^{PW(T)} = \prod_{n' \in \mathsf{child}(T,n)} N_{n'}^{PW(T)}$, if $\mathsf{kind}(n) = \underline{\mathsf{poss}}$

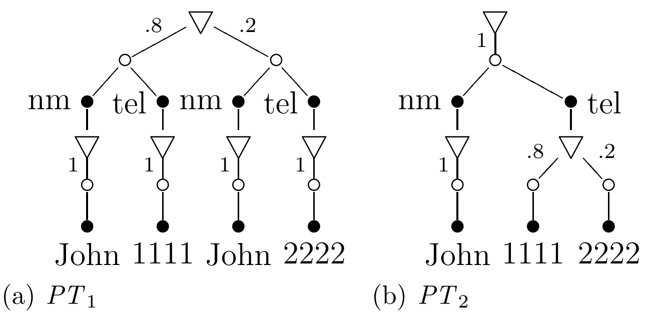


Figure 3.3: Probabilistic XML tree equivalence.

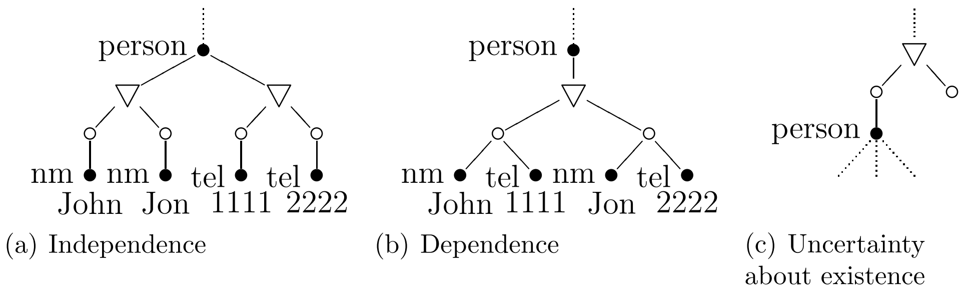


Figure 3.4: Expressiveness of probabilistic tree model.

- $N_n^{PW(T)} = \sum_{n' \in \mathsf{child}(T,n)} N_{n'}^{PW(T)}$, if $\mathsf{kind}(n) = \underline{\mathsf{prob}}$
- $N_n^{PW(T)} = \prod_{n' \in \mathsf{child}(T,n)} N_{n'}^{PW(T)}$, if $\mathsf{kind}(n) = \underline{\mathsf{xml}}$

Note that the above calculation gives the calculation for $|\mathsf{PWS}_{PT}|$.

Figure 3.3 shows an example of two equivalent probabilistic trees. They both denote the set of possible worlds containing trees with

- two nodes ‘nm’ and ‘tel’ with child text nodes ‘John’ and ‘1111’ respectively (probability .8) and
- two nodes ‘nm’ and ‘tel’ with child text nodes ‘John’ and ‘2222’ respectively (probability .2).

3.4 Expressiveness

As mentioned earlier, relational approaches often disallow dependencies among attributes. The higher expressiveness of the probabilistic tree makes such a restriction unnecessary. Figure 3.4 illustrates three common patterns: independence between attributes (Figure 3.4(a)), where any combination of ‘nm’ and ‘tel’ is possible. The advantage in XML is that values only have to be stored once, if they are independent of other elements or values. The second pattern is dependency between attributes (Figure 3.4(b)), where only the combinations 'John'/'1111' and 'Jon'/'2222' are possible. In this case the value of one element depends on the value of another element. The last pattern is uncertainty about the existence of an object (Figure 3.4(c)). Here one possibility is empty, i.e., has no subtree. The meaning of this empty subtree is not that the value is unknown, but rather that the subtree simply doesn't exist. These patterns can occur on any level in the tree, which allows a much larger range of situations to be expressed.

3.5 Trio data model

The Trio data model is based on the relational model. The Trio system is an Uncertainty and Lineage Database (ULDB) that captures uncertainty about the existence of data and also keeps track of where the data came from.

Uncertainty

The uncertainty in Trio uses Type-2 uncertainty. Each tuple in the database can be uncertain both in existence and appearance. Instead of regular tuples, alternatives for a tuple are stored and these alternatives are mutually exclusive. The set of alternatives is called an x-tuple. In addition, a tuple can be annotated with a questionmark, indicating that there is a possibility the tuple doesn't exist at all.

Example 1 *Table 3.2 shows an address book example in the Trio model. In this example, information about a person named John is stored. He either has room number 3035, or room number 3122. The room of a second person named Mary is either 3120, or 3110, or the entire tuple about this person doesn't exist. This possible non-existence of the tuple is indicated by the question mark.*

In this case no probability is associated with the alternatives, which indicates that alternatives are mutually exclusive, but no information is given about the relative likelihood of the alternatives.

Alternatives in Trio can have associated probabilities. Although it is possible to deviate from probability theory, the default is to adhere to probabilistic computations. This means, that the sum of probabilities associated with alternatives within one x-tuple does not exceed 1. If the sum is less than

Addressbook (name, room)	
(John, 3035) ‖ (John, 3122)	
(Mary, 3120) ‖ (Mary, 3110)	?

Table 3.2: Trio address book example

Addressbook (name, room)
(John, 3035):.8 ‖ (John, 3122):.2
(Mary, 3120):.6 ‖ (Mary, 3110):.2

Table 3.3: Trio address book example

1, this implicitly means that there is a questionmark on the tuple, making its existence uncertain.

Example 2 *We extend example 1 with probabilities on the alternatives to indicate relative likelihood of individual alternatives within x-tuples. The new address book is shown in Table 3.3. In the first x-tuple, the probabilities add up to* 1*, but in the second x-tuple, the probabilities do not add up to* 1 *and as a result, there is an implicit questionmark on the second x-tuple, indicating that the existence of the x-tuple itself is uncertain. The probability that this x-tuple does not exist is equal to remaining probability mass, here* 0.2*.*

3.6 Levels of Uncertainty

In an uncertain relational data model, there are several levels of uncertainty that can be distinguished. First, uncertainty can be associated with each tuple in the relation. This kind of uncertainty is shown in previous examples. The uncertainty at tuple level indicates if, and with which probability a tuple, or alternative, is present in the relation. This tuple level uncertainty is also referred to as Type-1 uncertainty [ZP97].

Another level of uncertainty is that associated with attributes. In this case, the tuple itself is certainly in the relation, but alternatives are specified at the granularity of attributes. This type of uncertainty is referred to as attribute level uncertainty, or Type-2 uncertainty [ZP97]. The information captured in Table 3.1(a) can be constructed by using either Type-1, or Type-2 uncertainty. The result is shown in Table 3.4. In the case with Type-1 uncertainty, there are 2 tuples, both with 2 alternatives, resulting in 4 tuples

(a) Type-1 representation

Addresses

Name	**Phone**	
John	555-1234	0.8
John	555-4321	0.2
James	555-5678	0.7
James	555-8765	0.3

(b) Type-2 representation

Addresses

Name	**Phone**	
John	555-1234	0.8
	555-4321	0.2
James	555-5678	0.7
	555-8765	0.3

Table 3.4: Data represented with either Type-1 or Type-2 uncertainty

with associated probabilities in the relation. In the Type-2 situation, there are just 2 tuples, where one of the attributes can have multiple values with associated probabilities.

A last level of uncertainty is that associated with a table. If a table is considered to hold information about the world, then objects present in the world, but missing in the database, can be seen as uncertainty about the real world. In this case, the database doesn't *cover* the entire world (with respect to the domain of the database). *Coverage* in that sense, can be seen as a third level of uncertainty. Of course, in the context of address books the notion of coverage doesn't make too much sense, since keeping track of the number of people we don't store in the database is probably more time consuming than just storing the actual data. However, when we consider data from sensors being stored in the database, the knowledge that some readings are missing in the database, because the sensors produce data at a higher rate than the database can store, can be useful. To apply coverage to our address book example, we could specify that we actually know 4 people. Combined with the fact that only information about 2 people is stored, the resulting coverage is 0.5.

In uncertain relational models, these three levels of uncertainty have to be treated differently. Not only is the semantics behind the uncertainty different for each of the levels, but also the way to implement them and provide functionality to store, query and manipulate the uncertainty differs for each of the levels. In probabilistic XML there is no real difference between the three levels of uncertainty mentioned before. Depending on the context node, the probability associated with a certain node can be regarded as being Coverage uncertainty, Type-1 (table-level), or Type-2 (attribute-level). This context node dependency is illustrated in Figure 3.5. The context possibility node in this figure is indicated by $\star$. A probability associated with this node for its descendents, is of type-1 uncertainty, whereas that same probability for its ancestors is of Type-2 uncertainty. If we only consider the tree underneath

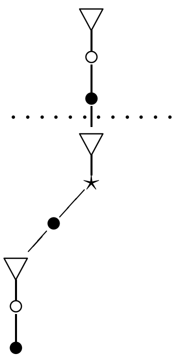


Figure 3.5: Context dependent levels of uncertainty

the dotted line, the probability indicates coverage.

3.7 DAG Representation

Although the compact representation is a lot more space efficient than the possible world representation, there still can be a lot of redundancy. In many cases, possible worlds have a lot of overlap, which can not be compacted using the compact representation. Consider the compact representation of the address book given in Figure 3.2. In this small example, the name of the person is repeated three times and both phone numbers are repeated twice. Sharing subtrees would be a logical solution. This turns the tree into a DAG..

3.7.1 Discovering Common Subtrees

The first step in constructing a DAG, is by going through the tree bottom-up. We construct buckets that contain a subtree from the leaf up. If two subtrees are equal, they point to the same bucket. As soon as in an iteration the subtrees differ, the bucket is considered a *common subtree* and the first occurrence of that tree is instantiated, while all next occurrences are references to the instantiation.

The result of this common subtree discovery phase, is a DAG, that can still contain duplicate information. This DAG is used as a starting point for further optimization.

3.8 Quality Measures

The measures we introduce in this section can be used for all data models, as long as local possibilities or alternatives can be identified. In IMPrECISE, our own probabilistic XML prototype that supports integration using uncertainty, probabilities are always local, because the probability associated with a possibility node expresses the likelihood of the subtree of that particular possibility node to hold the correct information about the real world. In relational systems such as Trio, probabilities are often associated with alternatives, which indicate the likelihood of an alternative being correct in the real world. This type of probability is also local. The number of choice points in IMPrECISE is equal to the number of probability nodes, since at each of these nodes a choice for one of the possibility nodes has to be made. In Trio the choice points are determined by the number of *x-tuples* in the relation. For each x-tuple one alternative has to be chosen.

We first define some notation. Let N_{cp} be the number of choice points in the data (i.e., probability nodes in IMPrECISE), $N_{poss,cp}$ the number of possibilities or alternatives of choice point cp, and let P_{cp}^{max} be the probability of the most likely possibility of choice point cp.

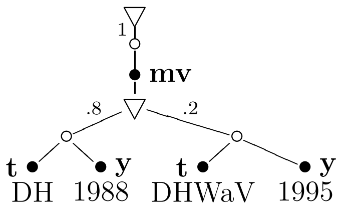
3.8.1 Number of possible worlds

An often used measure for the amount of uncertainty in a database is the number of possible worlds it represents. The number of possible worlds $|\mathsf{PWS}_{PT}|$ can be used to measure the amount of uncertainty present in the document. More uncertainty about individual objects, results in more possible worlds in the information source. The number of possible worlds is exponential to the number of objects described by the database.

3.8.2 Uncertainty density

The number of possible worlds, $|\mathsf{PWS}_{PT}|$ can be used as a measure for the amount of uncertainty in the document. This measure, however, exaggerates the perceived amount of uncertainty, because it grows exponentially with linearly growing independent possibilities. Furthermore, we would like all measures to be numbers between 0 and 1. We therefore propose the *uncertainty density* as a measure for the amount of uncertainty in a database. It is based on the average number of alternatives per choice point:

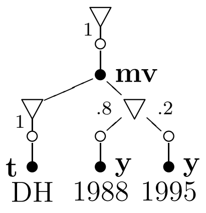
$$\mathrm{Dens} = 1 - \frac{1}{N_{cp}} \sum_{j=1}^{N_{cp}} \frac{1}{N_{poss,j}}$$



$N_{cp} = 2,$
$N_{poss,1} = 1, N_{poss,2} = 2$
$P_1^{max} = 1, P_2^{max} = .8$

$\text{Dens}= 1 - \frac{1}{2}(\frac{1}{1} + \frac{1}{2})$
$= \frac{1}{4} = .25$
$\text{Dec}= \frac{1}{2}(1 + \frac{.8}{1.2})$
$= \frac{5}{6} = .83$

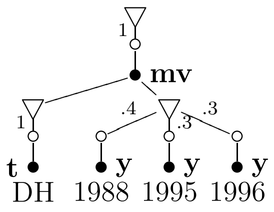
(a) Example A



$N_{cp} = 3,$
$N_{poss,1} = N_{poss,2} = 1,$
$N_{poss,3} = 2, P_3^{max} = .8$
$P_1^{max} = P_2^{max} = 1,$

$\text{Dens}= 1 - \frac{1}{3}(\frac{1}{1} + \frac{1}{1} + \frac{1}{2})$
$= \frac{1}{6} = .17$
$\text{Dec}= \frac{1}{3}(1 + 1 + \frac{.8}{1.2})$
$= \frac{8}{9} = .89$

(b) Example B



$N_{cp} = 3,$
$N_{poss,1} = N_{poss,2} = 1,$
$N_{poss,3} = 3, P_3^{max} = .4$
$P_1^{max} = P_2^{max} = 1,$

$\text{Dens}= 1 - \frac{1}{3}(\frac{1}{1} + \frac{1}{1} + \frac{1}{3})$
$= \frac{2}{9} = .22$
$\text{Dec}= \frac{1}{3}(1 + 1 + \frac{.4}{1.6 \times \log_2 3})$
$= \frac{1}{3}(2 + \frac{.4}{2.536}) = .72$

(c) Example C

Figure 3.6: Examples of uncertainty density and decisiveness

Dens is 0 for a databases that contains no uncertainty. Dens decreases if there is more certain data in the database for the same amount of uncertain data (compare Figures 3.6(a) and 3.6(b)). Dens rises if a choice point contains more alternatives (compare Figures 3.6(b) and 3.6(c)). If all choice points contain n alternatives, Dens is $(1 - \frac{1}{n})$, which approaches 1 with growing n. The uncertainty density is independent of the probabilities in the database. It can be used, for example, to relate query execution times to, because query execution times most probably depend on the number of alternatives to consider.

3.8.3 Answer decisiveness

Even if there is much uncertainty, if one possible world has a very high probability, then any query posed to this uncertain database will have one, easy to distinguish, most probable answer. We say that this database has a high *answer decisiveness.* In contrast, if there is much uncertainty and the probabilities are rather evenly distributed over the possible worlds, then possible answers to queries will be likely to have similar probabilities. We have defined the answer decisiveness as

$$\text{Dec} = \frac{1}{N_{cp}} \sum_{j=1}^{N_{cp}} \frac{P_j^{max}}{(2 - P_j^{max}) \times \log_2(max(2, N_{poss,j}))}$$

Dec is 1 for a database that contains no uncertainty, because each term in the sum becomes $\frac{1}{(2-1)\times\log_2 2} = 1$. If at each choice point j with two alternatives, there is one with a probability close to one (i.e., P_j^{max} is close 1), then all terms for j are also close to 1 and Dec is still almost 1. When P_j^{max} drops for some j, then Dec drops as well. Dec also drops when choice points occur with growing numbers of alternatives. This is accomplished by the $\log_2(\max(2, N_{poss,j}))$ factor (compare Figures 3.6(b) and 3.6(c)). We have taken the logarithm to make it decrease gradually.

3.8.4 Experiments

Set up

In this chapter we introduced the measures for uncertainty density and decisiveness. The purpose of the experiments hence is not to validate or compare systems or techniques, but an evaluation of the behavior of the measures to validate their usefulness.

name	**repr.**	**#pws**	**#nodes**
2x2	tree	16	469
4x4	tree	2,944	7,207
6x6	tree	33,856	25,201
6x9	tree	2,258,368	334,616
2x2 +rule	tree	4	328
4x4 +rule	tree	64	2,792
6x6 +rule	tree	256	8,328
6x9 +rule	tree	768	21,608
6x15 +rule	tree	3,456	87,960
2x2	dag	16	372
4x4	dag	2,944	1,189
6x6	dag	33,856	2,196
6x9	dag	2,258,368	13,208
2x2 +rule	dag	4	280
4x4 +rule	dag	64	761
6x6 +rule	dag	256	1,243
6x9 +rule	dag	768	1,954
6x15 +rule	dag	3,456	4,737

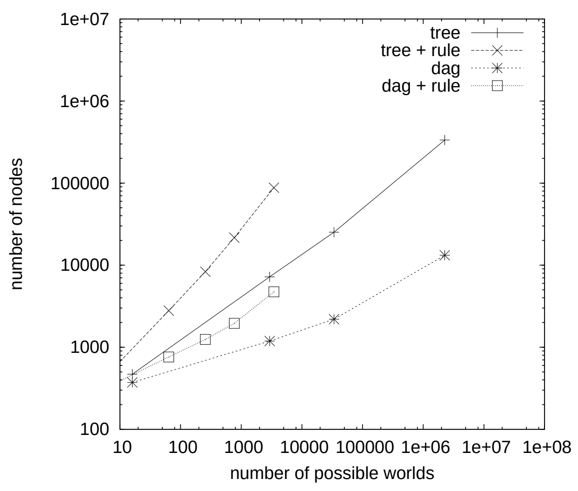
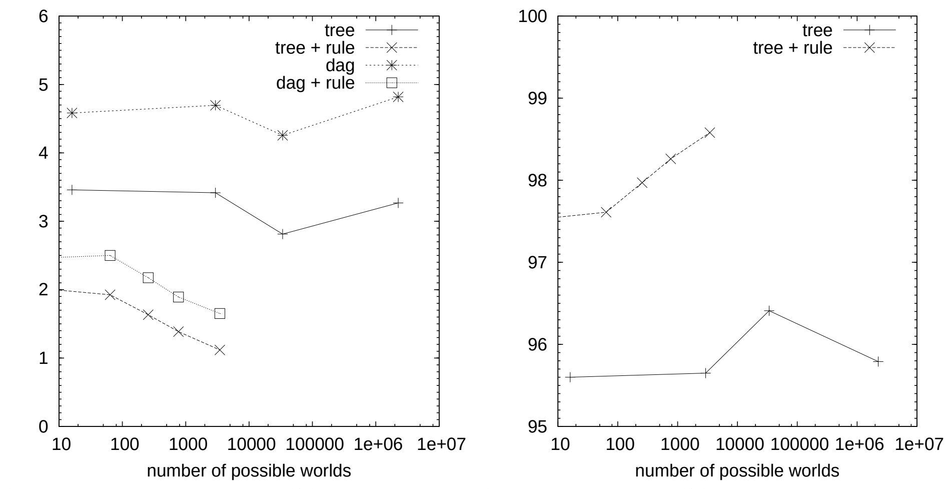


Figure 3.7: Data sets (pws = possible worlds)

As application of uncertainty in data, we selected data integration. In our research on IMPrECISE we attempt to develop data management functionality for uncertain data to be used for this application area. When data sources contain data overlap, i.e., they contain data items referring to the same real world objects, they may conflict and it is not certain which of the sources holds the correct information. Moreover, without human involvement, it is usually not possible for a data integration system to establish with certainty which data items refer to the same real world objects. To allow for unattended data integration, it is imperative that the data integration system can handle this uncertainty and that the resulting (uncertain) integrated source can be used in a meaningful way.

The data set we selected concerns movie data: Data set 'IMDB' is obtained from the Internet Movie DataBase from which we converted title, year, genre and director data to XML. Data set 'Peggy' is obtained from an MPEG-7 data source of unknown but definitely independent origin. We selected those movies from these sources that create a lot confusion: sequels, documentaries, etc. of 'Jaws', 'Die Hard', and 'Mission Impossible'. Since the titles of these data items look alike, the data integration system often needs to consider the possibility of those data items referring to the same real-world objects, thus creating much uncertainty in the integration result. The integrated result is an XML document according to the aforementioned probabilistic tree technique [KKA05].



(a) Uncertainty density (%) (b) Decisiveness (%)

Figure 3.8: Uncertainty density and decisiveness

To create integrated data sets of different sizes and different amounts of uncertainty, we integrated 2 with 2 movies selected from the sources, 4 with 4, 6 with 6, and 6 with 15 movies. We furthermore performed this integration with (indicated as '+rule') and without a specific additional rule that enables the integration system to much better distinguish data about different movies. This results in data sets with different characteristics. To be able to investigate uncertainty density, we additionally experiment with the data represented as tree as well as DAG. Although our implementation of the DAG representation does not produce the most optimally compact DAG yet, it suffices to experiment with its effect on uncertainty density. See Figure 3.7 for details of the data sets and an indication of the compactness of the representation.

Uncertainty density

Figure 3.8(a) shows the uncertainty density for our data sets. There is a number of things to observe.

- Density values are generally rather low. This is due to the fact that integration produces uncertain data with mostly choice points with only one alternative (certain data) and relatively few with two alternatives (uncertain data). For example, the '6x9 tree' case has 74191 choice points with one alternative and 5187 choice points with two alternatives.

- When comparing the lines for ‘tree’ with ‘dag’, and ‘tree + rule’ with ‘dag + rule’, we observe that the dag-versions have a considerable higher uncertainty density. This can be explained by the fact that the DAG representation shares common subtrees. Most commonality appears for certain data that occurs in all possible worlds. Hence, *relatively* more nodes are devoted to uncertainty in the DAG representation. The uncertainty density measure correctly exhibits this behavior.

- When comparing the lines for ‘tree’ with ‘tree + rule’, and ‘dag’ with ‘dag + rule’, we observe that the additional rule not only reduces the number of possible worlds, but also reduces the uncertainty density. The knowledge of the rule reduces uncertainty, but the amount of certain information stays the same. Therefore, it is logical that the uncertainty density goes down.

- The ‘+ rule’ lines drop with growing database size, while the other two do not. Database growth in this experiment means additional movies in both data sources. The specific rule we used in this experiment helps the integration system to determine which pairs of data items from both sources cannot possibly refer to the same real world object. The density measure correctly shows that the additional movies cause relatively more confusion without the rule than with it.

In general, we can say that important characteristics concerning the amount of uncertainty in the database can be assessed successfully with the uncertainty density measure. Moreover, it does not suffer from the disadvantage of exaggeration that the number of possible worlds has.

Answer decisiveness

Figure 3.8(b) shows the answer decisiveness for our data sets. This experiment focuses on the tree representation only, because the answers produced by a query is independent of the representation, hence the answer decisiveness does not depend on the representation. There are a number of things to observe.

- Decisiveness values are generally rather high. This has the same reason as why density is generally low: there are mostly choice points with only one alternative and few with two alternative, hence in most cases it is easy to make a choice for an answer because there is only one to choose from.

- Similar patterns in the lines for decisiveness can be observed when comparing with uncertainty density. Both measures are related, because the more alternatives per choice point on average, the higher the uncertainty density, but also the lower the decisiveness. Decisiveness only starts to deviate from density if the associated probabilities ensure that it is easy to choose the most likely possible answer. The probability assignment logic in our system, however, is still in its infancy and is apparently not capable of giving good decisiveness despite high uncertainty density.

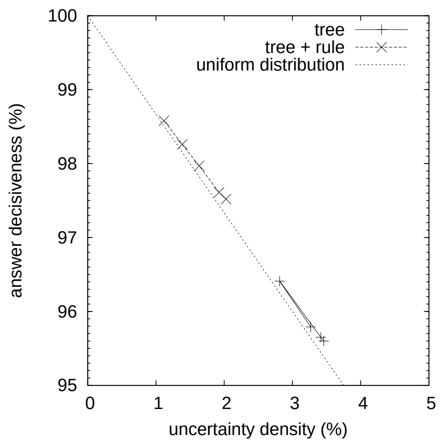


Figure 3.9: Density vs. Decisiveness

The relationship between the density and decisiveness measures is illustrated by Figure 3.9. The straight line marked 'uniform distribution' is drawn for the situation where the probabilities are always uniformly distributed and, for simplicity, where there are only choice points with at most two alternatives (which is the case for our test data and which makes the line straight). In this situation, uncertainty density fully determines answer decisiveness. The fact that the lines are not on the straight line shows that the probability assignment logic of our system has some impact on decisiveness despite the uncertainty density, but the impact is (as expected) rather limited. We expect that an integration system with better probability assignment logic will produce points much higher in the graph. Most importantly, the decisiveness measure can be effectively used to measure the quality of the probability assignment logic.

Chapter 4

Querying Uncertain Data

In this chapter we discuss and explain querying of uncertain data and in particular querying of probabilistic XML. The semantics of the query languages will be an important topic of interest, as will be the rewriting from probabilistic versions of the query language to non-probabilistic versions of query languages.

4.1 Semantics

As explained earlier, the semantics adopted in most uncertain and probabilistic databases in general and in IMPrECISE in particular, is that of possible worlds. The commutative diagram in Figure 4.1 shows how the semantics are followed.

Irrespective of the data model, the database D contains information, which represents a set of possible worlds, $D_1, \cdots, D_n$. The downward arrow on the left side in Figure 4.1 denotes the representation function PI. Operations q are, semantically, performed on this set of possible worlds (bottom-right arrow). The result $R^q(D_1), \cdots, R^q(D_n)$ can, of course, be transformed back into one single database $R^q(D)$ (upward arrow on the right). The implementation of operations uses the direct connection (right arrow at the

$$\begin{tikzcd}
D \arrow[r, "Q"] \arrow[d, leftrightarrow, "PI"] & R^q(D) \arrow[d, leftrightarrow, "PI"] \\
D_1, \cdots, D_n \arrow[r, "q"] & R^q(D_1), \cdots, R^q(D_n)
\end{tikzcd}$$

Figure 4.1: Commutative diagram

top). In order words, the bottom part of the figure represents the semantics of querying, the possible worlds, whereas the top of the figure represents the operational semantics, or the compact representation and operations working directly on the compact representation without enumerating all possible worlds.

In case a query is posed to a database using the possible world semantics, this means that the query is evaluated in every world represented by the database. The results from each of the worlds is collected and the union of the results from all worlds is presented to the user as the result of the query to the database. In many cases, different worlds give the same query result which can be merged and their probabilities added, but this is merely a presentational improvement. The idea behind the possible world approach is that all worlds have a part in the result, proportional to their associated probability.

Example 3 *Consider a document containing movies. Obtaining all titles from movies released in the year* 1995 *using probabilistic XPath is equal to evaluating the non-probabilistic XPath version on each of the possible worlds. The query would be*

```
/movies/movie[./year='1995']/title
```

The result of this query is constructed by evaluating the query in each of the possible worlds and combining the results of these worlds into one.

4.2 Relational querying

In Trio, queries are transformed from the Trio Query Language (TriQL) to a SQL statement that is passed to the underlying database. TriQL itself is a SQL like language. Statements are of the form SELECT FROM WHERE, but instead of querying a regular database, possible worlds are queried in the way described in the previous section. The result is then grouped by x-tuple, such that all alternatives for each x-tuple end up in the same x-tuple again. For regular SELECT queries this means that a simple ORDER BY statement is added to the query that is passed to the underlying database, sorting the result by x-tuple. The Trio layer splits the result on changing x-tuples and presents the result to the user on a per x-tuple basis.

In case of SELECT DISTINCT queries, this method becomes somewhat more complicated, because now duplicates in, but also between x-tuples have to be eliminated. Two queries are needed to perform a SELECT DISTINCT in Trio. The first query eliminates duplicates, while the second query reorders the result on x-tuple id.

$$\begin{tikzcd}
D \arrow[r, dotted, "Q"] \arrow[d, leftrightarrow, "PI"] & R^q(D) \arrow[d, leftrightarrow, "PI"] \\
D_1, \cdots, D_n \arrow[r, "Q"] & R^q(D_1), \cdots, R^q(D_n)
\end{tikzcd}$$

Figure 4.2: Extended commutative diagram

4.3 XPath queries

In IMPrECISE queries are transformed into XQuery. We did not yet work on the implementation of querying compact or DAG representation directly as a result, the database is expanded to the possible world representation and the query is executed in every possible world, exactly as prescribed by the semantics. The results are combined and presented to the user.

As a result, the current implementation returns a sequence with certain XML elements with an associated probability of the world that produced that particular element as a result to the query. Preliminary results do show that in case of forward axes, rewriting a query that can be used directly on the compact representation is straightforward. Adding backward axes and predicates to be used directly on the compact representation, and especially the DAG representation is future research.

4.4 Across Possible Worlds

Although the semantics used in IMPrECISE is that of possible worlds, there are two language constructs that don't follow the commutative diagram from Figure 4.1 and therefore don't adhere to the possible world approach as explained in Section 4.1. These language constructs are *horizontal queries* and all of the new *aggregates*.

4.4.1 Horizontal Queries

In many cases comparing possible worlds, or choosing between possible worlds, is convenient, or even necessary. A first example of such a query could be:

Retrieve for each movie title, the alternative with the highest probability.

This would require a comparison between possible worlds, since it is not necessarily true that one possible world contains all *most likely* movie-alternatives. Another query is

Title	Year	Rating	
Die Hard	1988	4	.33
	1990	3	.33
	1995	3	.33
King Kong	1933	4	.05
	1976	3	.2
	2005	4	.75

Table 4.1: Movie Database Example

Retrieve all movies that have at most 3 *alternatives*

This query selects all movies where the number of alternatives is low (at most 3), which causes a high decisiveness for that particular query.

The construct to query between possible worlds in Trio, is called *horizontal subquery*[1]. The alternatives of one element or tuple are viewed as a separate relation. This relation is then queried as a *normal* query, where the alternatives are treated as tuples in this new mini relation. This mini relation represents sets of possible worlds instead of the possible worlds separately, this is due to the fact that a mini relation restricts the database to only one object. Objects have alternatives, that do not correspond to the possible worlds directly, but to sets of possible worlds. Therefore, a horizontal subquery doesn't follow the commutative diagram given in Figure 4.1, because after querying the possible worlds, the answers are not directly presented as the result. In case of querying probabilities, the answer is not even obtained from querying possible worlds. The extended commutative diagram in Figure 4.2 does capture constructs working across possible worlds, by introducing an *operational* arrow from the original database to the result set, indicated by the dashed arrow in the figure.

In Trio, horizontal subqueries are surrounded by square brackets. The syntax of a horizontal subquery is identical to the syntax of regular SQL.

Example 4 *Consider a movie database with schema* $movie(title, year, rating)$ *as shown in Table 4.1. All of the attributes are uncertain, i.e. can have alternatives. The rating column holds a number between 0 and 5 indicating the user rating for that particular movie. We pose a query to retrieve only those tuples that have a high decisiveness, i.e. the number of alternatives is*

[1]The term horizontal subquery was introduced in the Trio project at Stanford University

Title	Year	Rating	CONF
King Kong	1933	4	.05
	1976	3	.2
	2005	4	.75

Table 4.2: Movie Database Result

at most 3 and at least one of these alternatives has an associated probability of 75% or more. The resulting query is:

```
SELECT *
FROM movie
WHERE EXISTS [SELECT *
              FROM movie
              WHERE count(*)<4 AND MAX(CONF) >= 0.75]
```

The reference to movie in the horizontal subquery is just to be compatible with the SQL standard and can be read as a reference to the current *x-tuple from the movie table being evaluated. The result of the query is given in Table 4.2.*

We introduced some abbreviations for horizontal subqueries, because they are mostly used in very specific settings. Horizontal subqueries mostly query those tables that are also used in the outer query. The first abbreviation omits the FROM clause altogether, assuming the same list of tables as used in the outer query. Also, most often horizontal subqueries are used in the WHERE clause as part of an EXISTS statement. In these cases, not even the select statement is needed. The second abbreviation leaves out the FROM statement and the SELECT statement, reducing the horizontal subquery to a WHERE clause. The keyword WHERE in this case is also omitted. The previously given query can be rewritten using the second abbreviated form as follows.

```
SELECT *
FROM movie
WHERE EXISTS [count(*)<4 AND MAX(CONF) >= 0.75]
```

4.4.2 Aggregates

Aggregate functions combine the values of a set of attribute values into one value. Examples are *the total price of a collection* or *the average mark of* *students.* Observe that a traditional aggregate operator works in *one* world where everything is certain. To come up with a proper semantics for aggregate operators in the context of probabilistic relations, we use the strategy of enumerating possible worlds. For each possible world pw with associated probability $P(\mathsf{pw})$, we apply the traditional aggregate operator $aggr$. The results $(aggr, P(\mathsf{pw}))$ together form the resulting probabilistic relation. Observe that normal aggregates do not work across possible worlds.

Given the movie rating relation of Table 4.2 and the query below:

```
SELECT MAX(Rating)
FROM movies
```

This would traditionally return the number 4. This is, however, only true for the worlds where 'King King' is produced in 1933 or 2005, but there is another possible world. This possible world should contribute to the result. The correct result

$$\begin{aligned} & \{(4, 0.05), (3, 0.20), (4, 0.75)\} \\ = \; & \{(3, 0.20), (4, 0.80)\} \end{aligned}$$

reflects the existence of three possible worlds, one possible world for each of the possible ratings for "King Kong". The final answer reflects the fact that two of those worlds have the same aggregate result and are therefore combined and their associated probabilities added.

Let pR be a probabilistic relation, for example a Trio relation and let pT be a probabilistic tuple. In more general terms, the probabilistic aggregate operators (MAX, MIN, SUM, AVG) are defined by

$$aggr_f(pR) \in \mathbb{P}([0, 1] \times \mathbb{R})$$

where f indicates a field name. We use the notation $\overline{aggr}_f(\mathsf{pw})$ for the traditional counterpart of an aggregate operator evaluated in possible world pw. The aggregate function $aggr_f$ is defined by

$$aggr_f(pR) = \{(P(\mathsf{pw}), \overline{aggr}_f(\mathsf{pw})) \mid \mathsf{pw} \in \mathsf{PWS}\}$$

EXP function

All aggregate operators take the existence of possible worlds into account. However, the system should be able to predict information about the real world. We, therefore, introduce a new aggregate function, *EXP*, which returns the expected value of a numerical field. The *EXP* aggregate function, therefore, does work across possible worlds, since it computes an aggregate over the worlds.

$$EXP_f(pR) \in \mathbb{R}$$

It is defined by

$$\frac{\sum_{pT \in pR} \pi_f(pT) \times P_f(pT)}{\sum_{pT \in pR} P_f(pT)}$$

Where $\pi_f(pT)$ is the value of attribute f in pT. The function calculates the weighted average of the field over all possible worlds. In the presence of a GROUP BY clause, we assume *pR* to represent *one group*. For example,

```
SELECT EXP(Rating)
FROM movies
GROUP BY title
```

returns $\frac{4\times0.05+3\times0.20+4\times0.75}{1.0} = \{3.8\}$, the expected ratings for all movies in the relation.

EXP can be used in combination with other aggregate functions. The expected maximum rating is obtained by

```
SELECT EXP(MAX(Rating))
FROM movies
```

The fact that *EXP* works across possible worlds, can be clearly observed in the above example. The MAX aggregate results from all possible worlds are passed to the *EXP* function. By using results from different possible worlds, the *EXP* function works across possible worlds.

4.4.3 Querying Probabilities

Within a world, the probability for that world is not known, i.e. using normal database operations, the probability for a world is inaccessible. In this respect, querying probabilities, or uncertainty in general, i.e. the number of alternatives for an element, is an operation that works across possible worlds. New functions have to be defined to query the probability and a construct is needed to compare between possible worlds.

In Trio, for example, the CONF function returns the probability for an alternative. However, when using this function, the user should take notice of

title	
Die Hard I	0.8
Die Hard II	0.7
King Kong	0.9

Table 4.3: Movie series

the exact meaning of the result. The function can either return the probability of a specific alternative, or the probability of the return value, which is in general the sum of the possible worlds where this particular value exists. In many cases these probabilities are the same, but there are some exceptions. These exceptions occur when data dependency arises.

Example 5 *Consider the Table 4.3 which contains titles of movies. We pose a query to get those combinations of movies that possibly belong to a series. In this query, the function distance(a, b) returns the edit-distance of parameters a and b.*

```
SELECT a.title, b.title, CONF(a.title), CONF(b.title), CONF(*)
FROM movies a, movies b
WHERE (distance(a.title, b.title) < 3) AND
      (a.title < b.title)
```

The result of this query is ("Die Hard I", "Die Hard II", 0.8*,* 0.7*,* 0.56*) with probability* 0.56*. Probabilities* 0.8 *and* 0.7 *in the original table are the alternative based probabilities, however, the probability associated with the result alternative is* 0.56*. The original probabilities are retrieved by using CONF(title), while the result probability is obtained with CONF(*).*

4.5 Updates

An update in the possible world approach, means performing the update in every possible world. If an update is performed, where the value of an element, or elements is changed and as a result, the element becomes equal to an already existing element, the number of possible worlds and hence, the amount of uncertainty is reduced.

Example 6 *Consider Table 4.2 containing title, year and rating for movies and the following update*

Title	Year	Rating	CONF
King Kong	1976	3	.2
	2005	4	.80

Table 4.4: Movie Database Result

```
UPDATE movies SET Year=2005 WHERE Year=1933
```

The result of the update is shown in Table 4.4. This table shows that the number of alternatives is reduced from 3 to 2, because the year 1933 changed to 2005 and all other attribute values of the 1933 alternative were already equal to the values of the 2005 alternative, merging the two alternatives in the updates databases. The associated probabilities of the two alternatives were added.

When the data source is considered to be the result of multiple sources that are integrated, feedback on this source can be applied to indicate the validity of the integrated data. This feedback process is discussed in Chapter 6. Note that feedback is *not* equal to performing an update on the document.

4.6 Answer Quality

Querying uncertain data results in answers containing uncertainty. Therefore, an answer is not correct or incorrect in the traditional sense of a database query. We need a more subtle notion of answer quality.

In the possible world approach, an uncertain answer represents a set of possible answers each with an associated probability. In some systems it is possible to work with alternatives without probabilities, but these can be considered as equally likely, hence with uniformly distributed probabilities. The set of possible answers ranked according to probability has much in common with the result of an information retrieval query. We therefore base our answer quality measure on precision and recall [BYRN99]. We adapt these notions, however, by taking into account the probabilities of the possible answers. Correct answers with high probability are better than correct answers with a low probability. Analogously, incorrect answers with a high probability are worse than incorrect answers with a low probability.

XQuery answers are always sequences. The possible answers to an XQuery on an uncertain document, however, largely contain the same elements. Therefore, we construct an amalgamated answer by merging and ranking the elements of all possible answers. This can be accomplished in XQuery

```
declare function rank_results($pws as element(world)*)
  as element(answer)*
{
  for $v in distinct-values($pws/descendant::text())
  let $ws := $pws[./descendant::text()[.=$v]]
    ,$rank := sum($ws/@prob)
  order by $rank descending
  return <answer rank="{$rank}">{$v}</answer>
};
```

Figure 4.3: XQuery function for ranking query results

with the function in Figure 4.3. The effectiveness of this approach to querying a probabilistic database can be illustrated with an example. Suppose we query a probabilistic movie database asking for horror movies:

`//movie[.//genre="Horror"]/title`.

Even though the integrated document may contain thousands of possible worlds, the amalgamated answer is restricted to the available movie titles considered to be possibly belonging to a horror movie, which will be few in number.

Precision and recall are traditionally computed by looking at the presence of correct and incorrect answers. Let H be the set of correct answers to a query (as determined by a human), A the set of answers (the elements of the amalgamated query answer), and C the intersection of the two, i.e., the set of correct answers produced by the system (see Figure 4.4).

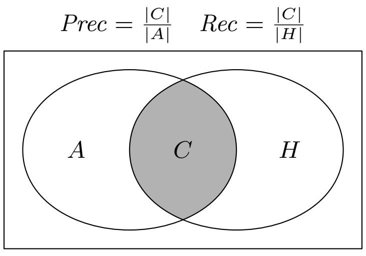


Figure 4.4: Precision and recall.

We adapt the precision and recall measures by taking into account the probabilities: An answer a is only present in the amount prescribed by its probability $P(a)$. Incorrect answers, on the other hand, are also only present in the amount prescribed by their probabilities. Incorrect answers are those answers present in the answer set A, but not present in the correct document set H, or $a \in (A - C)$

This reasoning gives us the following definitions for precision and recall.

$$\text{Prec} = \frac{\sum_{a \in C} P(a)}{|C| + \sum_{a \in (A-C)} P(a)} \qquad \text{Rec} = \frac{\sum_{a \in C} P(a)}{|H|}$$

Example 7 *Say the answer to the query "Retrieve all horror movies" is "Jaws" and "Jaws 2". If the system returns this answer, but with a confidence* *of 90% for both movies, then precision and recall are both* $\frac{0.9+0.9}{2} = 0.9$*. If, however, it also gives some other (incorrect) movie with a confidence of 20%, then precision drops to* $\frac{0.9+0.9}{2+0.2} = 0.82$ *and recall stays* 0.9*.*

4.6.1 Experiments

To obtain test data suitable for evaluating our answer quality measure, we took one of the data sources: an IMDB document with 9 movies. We made two copies of it, randomly polluted them by corrupting text nodes, and then integrated them. We made sure we didn't pollute the same text nodes, so 'the truth' is still available in the combined data of both sources and an ideal integration system would be able to reconstruct it. We furthermore took three queries and posed them to the data integration result of data sources with increasing pollution. A pollution of 2 means that 2 randomly chosen text nodes in both sources have been corrupted by changing a randomly chosen character to '@'. This pollution not only affects the data integration, also in some of the answers we see these modified strings appear. Although they are seemingly almost correct, we classified these answers as incorrect.

Table 4.5 shows the answer quality measurements for the three queries. In this table an "X" behind the probability indicates that the answer is incorrect. Even though our system produces the correct answers in most cases, the confidence scores the system produces are rather modest. This is due to the naive probability assignment explained earlier. Our adapted precision and recall measures effectively reflect this aspect of reduced answer quality. Missing answers (as in Query 1 / Pollution 20, and Query 3 / Pollution 20) is of course worse than just modest confidence scores; indeed radically lower recall is given to these cases.

(a) Query 1: `//movie[.//genre="Horror"]/title` (All horror movies)

Poll	P(a)		Answer	Prec	Rec
2	79.4%		“Jaws”	79.4%	79.4%
	79.4%		“Jaws 2”		
5	77.4%		“Jaws”	69.5%	77.4%
	77.4%		“Jaws 2”		
	22.6%	X	“Ma@ing of Steven Spielberg’s ’Jaws’, The”		
10	85.4%		“Jaws”	74.5%	85.4%
	85.4%		“Jaws 2”		
	29.2%	X	“Ma@ing of Steven Spielberg’s ’Jaws’, The”		
20	85.4%		“Jaws”	74.5%	42.7%
	14.6%	X	“Ma@ing of Steven Spielberg’s ’Jaws’, The”		

(b) Query 2: `//movie[./year="1995"]/title` (All movies produced in 1995)

Poll.	P(a)		Answer	Prec	Rec
2	100.0%		“Die Hard: With a Vengeance”	100.0%	100.0%
	100.0%		“Behind the Scenes: Die Hard - With a Vengeance”		
	100.0%		“Making of Steven Spielberg’s ’Jaws’, The”		
5	79.4%		“Die Hard: With a Vengeance”	56.3%	64.3%
	58.8%		“Behind the Scenes: Die Hard - With a Vengeance”		
	54.8%		“Making of Steven Spielberg’s ’Jaws’, The”		
	20.6%	X	“Behind th@ Scenes: Die Hard - With a Vengeance”		
	11.3%	X	“Ma@ing of Steven Spielberg’s ’Jaws’, The”		
	5.6%	X	“Jaws”		
	5.6%	X	“Jaws 2”		
10	85.4%		“Die Hard: With a Vengeance”	47.1%	56.3%
	41.7%		“Behind the Scenes: Die Hard - With a Vengeance”		
	41.7%		“Making of Steven Spielberg’s ’Jaws’, The”		
	21.9%	X	“Behind th@ Scenes: Die Hard - With a Vengeance”		
	14.6%	X	“Ma@ing of Steven Spielberg’s ’Jaws’, The”		
	7.3%	X	“Jaws”		
	7.3%	X	“Jaws 2”		
	7.3%	X	“Die Hard 2”		
20	78.1%		“Die Hard: With a Vengeance”	52.6%	53.8%
	41.7%		“Behind the Scenes: Die Hard - With a Vengeance”		
	41.7%		“Making of Steven Spielberg’s ’Jaws’, The”		
	7.3%	X	“Behind th@ Scenes: Die Hard - With a Vengeance”		

(c) Query 3: `//movie[./title="Jaws 2"]/year`
(When was Jaws 2 produced?)

Poll.	P(a)		Answer	Prec	Rec
2	69.1%		“1978”	62.6%	69.1%
	10.3%	X	“1975”		
5	66.1%		“1978”	59.4%	66.1%
	5.6%	X	“1975”		
	5.6%	X	“1995”		
10	78.1%		“1978”	72.8%	78.1%
	7.3%	X	“1995”		
20	78.1%	X	“197@”	0.0%	0.0%
	7.3%	X	“@995”		

Table 4.5: Answer quality (‘X’ marks an incorrect answer)

Chapter 5

Information Integration

The previous chapters explained uncertain data and in particular probabilistic XML and querying uncertain data. This chapter focuses on one application of uncertain data, information integration.

We sketch the whole process of information integration including several techniques that can be used, starting from schema integration. However, the focus of this chapter is on the integration of the actual data in the presenence of overlapping data. We will first give an overview of the whole integration process.

5.1 The Process

Integration of information is accessing different sources of data, containing possibly overlapping information, as one single source. The different sources of information may, and probably will, have different schemas to represent the data.

When two, or more documents are integrated, several steps have to be taken sequentially. Without loss of generality, we will restrict ourselves to integration of two documents.

The first phase of information integration, is that of schema matching. Mappings between the schemas of the two source documents have to be established in order to determine which of the schema elements from both documents are semantically similar. These mappings can be from one schema element to one other schema element, but also more difficult mappings, such as m to n mappings are possible [Vis07]. The problem in finding these mappings, is the semantical meaning of the schema, not specifically expressed in any way in the schema itself.

The second step in the integration process, is combining the aligned schemas into one schema that will be used for the integrated document. Together with the new schema, transformation rules are generated to transform data from the schema of the original data sources to the schema of the new data source. The data from both sources is converted to the new schema.

The third step is to identify duplicates in the data. Every element from the first data source is compared with every element from the second data source. If a match is found, the two elements are merged into one resulting element, which is stored in the integrated document. If no duplicate is found, the elements are stored in the integrated document unaltered.

5.2 Kinds of Integration

We can distinguish different kinds of integration. All are concerned with transforming data from one schema to another, merging together data from different sources. The differences between the kinds of integration are usually due to prior knowledge of the source documents, or the relation between the source documents.

- Synchronization
  If one object is contained in more than one information source, after synchronization the descriptions of the object in all information sources have to be in agreement.

- Information Integration
  In this thesis, we only consider mediated integration (Figure 5.1(a)) In this case, multiple information sources, containing possibly overlapping information are presented as one global information source. The information sources are not actually integrated, but the global information source is virtual. A component, called a *mediator*, acts as the global information source. This mediator transforms queries to the global information source into queries to the underlying information sources and gathers and combines the results from these information sources, which can be presented as the result from the global information source. The underlying information sources in this case remain autonomous.

- Schema Evolution
  Schema evolution (Figure 5.1(b)) is accommodated when a database system facilitates the modification of the database schema without loss of existing information. When the schema of the database changes, information already contained in the database, can still be accessed, using the new schema. In the case of evolution, there is a distinction

between materialized and mediated evolution. In the case of materialized, the data is converted to the new schema, whenever the schema changes. Data is always stored using the current schema. In case of mediated evolution, the data is stored using the schema used to insert the data into the database. When a query is passed to the database, the mediator transforms the schema to match the schema of the data.

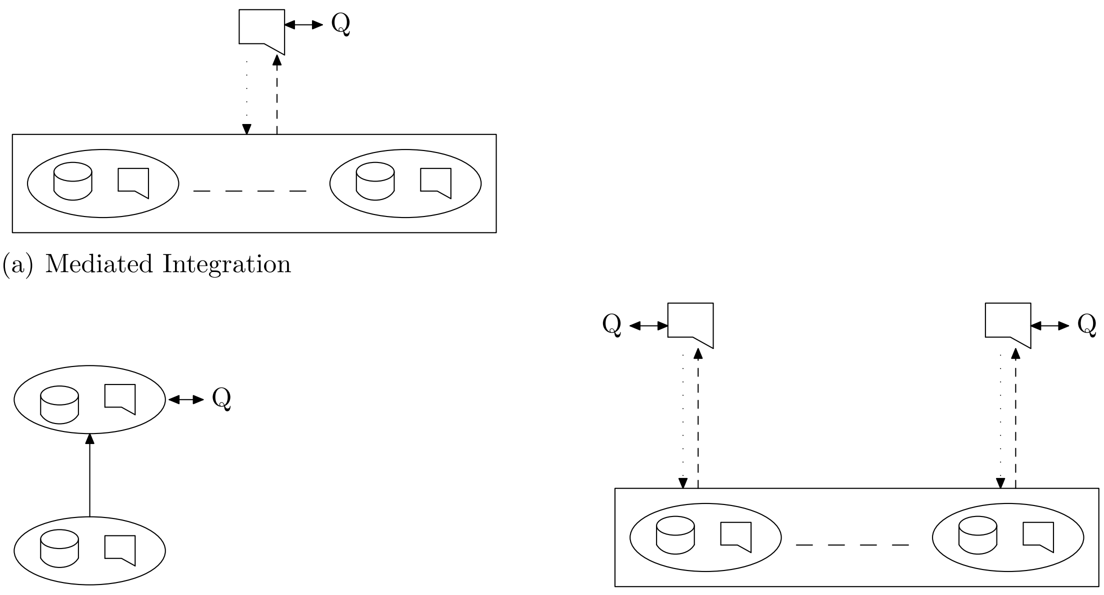
- Schema Versioning
  Schema versioning (Figure 5.1(c)) is accommodated when a database system allows access to all data, both retrospectively and prospectively, through user definable version interfaces. The data is stored in the database using the newest schema at the time the data was inserted into the database. Queries to the database can use any of the schemas ever used in the database. A mediator will have to transform the schema to match that of the data.

The latter two definitions are taken from and further explained in [Rod95]. Evolution and versioning can be seen as specializations of information integration. In all cases, data stored using different schemas are presented as having one schema. With integration, two or more information sources are combined into one source. Evolution allows users of the data to access old data through new schemas, demanding for integration of, at least, schemas of old and new data source. Versioning allows the user to access the data through any of the old or new schemas.

5.3 Integration Architecture

In this section, we show the integration architecture of the IMPrECISE system and explain the functionality of each of the components. A schematic overview of the system is given in Figure 5.2. Bottom-up, we identify the following components

- Datasources
  The underlying datasources that need to be integrated.
- Wrappers
  Provide a generic interface to the underlying datasource.
- Mediator
  The mediator is the central part of the integration system. It acts as a single datasource, i.e. mediated datasource, to any user (application or human) using the integration application. It consists of the following parts:



(a) Mediated Integration

(b) Evolution

(c) Versioning

Figure 5.1: Schematic view of integration concepts

- Schema conversion
  The rule engine in IMPrECISE makes explicit correspondences between the mediated schema and the schemas of the underlying data sources.
- Integrator
  Integrates the actual data values from the underlying information sources, using the Oracle to make semantical decisions.
- The Oracle
  Determines to what extent two tuples from the underlying information sources refer to the same *real world object*. The Oracle is responsible for decisions on semantical equivalence of elements and uses a set of rules to decide on this equivalence.

- Application
  An external application using the integration application as a DBMS.

5.4 Schema Integration

The integration of information starts at schema level. As shown earlier, schemas of the original information sources have to be transformed in order for the data to be integrated. The transformation of these schemas is preceeded by a matching of schemas, i.e. what parts of schema 1 match with

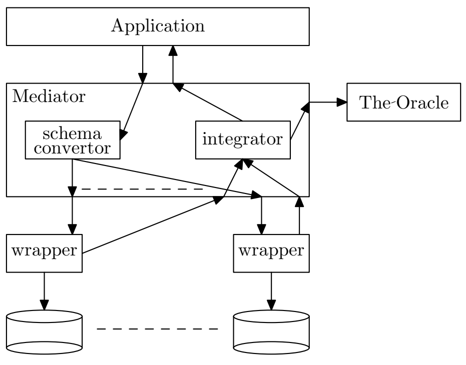


Figure 5.2: Schematic overview of IMPrECISE

what parts of schema 2. The methods used to match these schemas depend on the kind of schemas used in the information sources. Also, if instead of integration of data sources, evolution or versioning is used, matching will most likely be easier, since both schemas originate from the same ancestor schema.

The process of schema integration is discussed in section 5.4.1. Wrappers are an important aspect of integration and these will be the topic of section 5.4.2. In section 5.4.3 we will discuss the role of mediators and the way in which they can be designed. In section 5.4.4, matching techniques are discussed, taking into account the different types of schemas we can encounter. Using the notion of time is helpful with evolution and versioning, this will be the topic of section 5.4.6.

After finding mappings between elements, based on different methods, the results of all method/mapping combinations is aggregated to find the most likely matches. In [dV06] a framework is proposed that allows the system to learn from previous matches. At the end of the matching phase, the user is given the opportunity to give feedback on the matching result. The result is stored in a *Global Intermediate Schema*, which is used in subsequent runs of the schema matcher. As a result, these subsequent runs of the matcher give an increase in matching result. Most schema matchers only consider one to one matches. In [Vis07] a method is presented to find complex, n to m matches, based on word frequencies in the instance data. After schema elements have been matched, conversion functions are created to convert from one schema to another. For one to one matches, these conversion functions usually are trivial, this is certainly not the case for complex matches [Vis07].

5.4.1 Integration process

The process of integrating different schemas into one global schema [SL90], can be divided into four different phases [BLN86]:

- Pre-integration
  During this phase, the integration method is chosen and additional information is gathered.
- Comparison
  Source schemas are compared and similarities and differences are detected
- Conformation
  If conflicts in the schemas are detected, an attempt is made to resolve these conflicts.
- Merging and restructuring
  The results of the conformation phase are merged and restructured to ensure minimality and understandability. The result of this phase is an integrated and deduplicated integrated document.

5.4.2 Wrappers

Wrappers provide a common interface to a data source. If we have a data source A and a data source B, that both have different schemas, then wrappers translate the interaction from the user, specified in using the global schema, to the schema of the data source. Consider the two data sources shown in Table 5.1. The schema which the user uses to pose queries could be $(name, room, email)$. In this case, firstname and name from data source A should be mapped to name in the global schema and building and room in data source B should be mapped to room. But wrappers should also be able to map the other way, i.e. from user schema to actual data source. In this case, the email address is only stored in Table 5.1(a) , whereas objects from Table 5.1(b) do not have any email address. The integrated document, therefore, would give email addresses for objects found at least in Table 5.1(a) , but leave this attribute as a NULL value for objects only found in Table 5.1(b).

Besides converting between schemas, a wrapper is also used to convert between data models. If the global information source is presented as an XML source, underlying information sources can still be relational, object-oriented, or any other kind of information source. The function of the wrapper is to convert these relational and other kinds of information to equivalent data in the XML schema.

(a) Data source A

name	firstname	room	email
Doe	John	ZI-3122	john@doe.com
King	Ed	ZI-2012	ed@king.edu

(b) Data source B

name	building	room	phone
John Doe	ZI	3122	4243
Ed King	ZI	2012	3519

Table 5.1: Two semantically similar data sources with different schemas

In short, the function of a wrapper is to provide a common interface to the underlying data source. The mediated schema in Figure 5.3 can access all data sources through a common interface provided by the wrappers in between the mediator and the actual data source.

Designing these wrappers for various data sources is a tedious and time consuming process. In the TSIMMIS project [CGMH$^+$94, GMPQ$^+$97] wrappers are created semi-automatically.

5.4.3 Mediators

A mediator acts as integrated information source for several independent autonomous information sources (see Figure 5.3). Queries passed to the mediator are translated into queries to underlying information sources and the resulting information is combined and presented as result of the mediated information source. The underlying information sources are usually accessed through wrappers.

The TSIMMIS system [GMPQ$^+$97, GMHI$^+$95] is a mediated system. The system uses the Object Exchange Model (OEM) to query data.

Mediation can be looked at from different perspectives. The first way to mediate is Global as View, or GAV. With GAV, the mediated schema is a view on the local, autonomous data sources. The view itself does not contain any data, but is merely a query rewriting mechanism onto the underlying sources.

The second way to mediate is using Local as View, or LAV. Here the different underlying autonomous data sources are treated as views of the mediated schema. The views in, this case, contain the data. We will elaborate on both GAV and LAV.

Object Exchange Model

The mediator distributes the query over the different available information sources. The mediator can do this using a uniform schema kind. The main focus of the mediator in case of a query is distributing the query over the information sources itself, collecting the different parts of the information from all information sources.

In the TSIMMIS system this uniform schema is the Object Exchange Model (OEM) [GMPQ+97, PGMW95]. Data in OEM is *self-describing*, i.e. it contains its own schema and does not need an additional schema. Objects in the model consist of four parts. They have an *object ID*, and a *label*, which describes what the object represents. Labels are human readable and therefore contain all information needed about the object. The third part is a *type* of the value. It can either be an atomic type or *set*. The last part of an object is the *value*, which is either an atomic value, or a set of objects, depending on the type for that particular object.

Our address book example from Table 5.1(a) in OEM would look like

```
<set-of-addresses, set, {ad_1, ad_2}>
  ad_1: <name-and-address, set, {fn_1, ln_1, room_1, mail_1}>
  ad_2: <name-and-address, set, {fn_2, ln_2, room_2, mail_2}>
    fn_1: <firstname, string, 'John'>
    ln_1: <name, string, 'Doe'>
    room_1: <room, string, '3122'>
    mail_1: <email, string, 'john@doe.com'>
    fn_2: <firstname, string, 'Ed'>
    ln_2: <name, string, 'King'>
    room_2: <room, string, '2012'>
    mail_2: <email, string, 'ed@king.edu'>
```

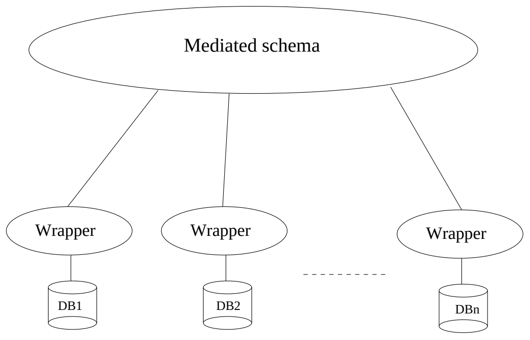


Figure 5.3: Mediator

Using such a uniform schema, requires the system to have knowledge of just one schema kind. Wrappers will transform data from the source schema to this uniform schema and back.

Global as view

With Global as View, the mediated information source is viewed as a view on the original information sources. Suppose we have three information sources, addressbook (name, room, phone), addresses (name, room) and phones (name, phone). In all sources attribute *name* is key. We want to create a mediated information source which provides the same information as initial address book example. We can compose a mediated view completebook as follows

```
CREATE View completebook AS
  SELECT * FROM addressbook

  union

  SELECT name, room, phone
  FROM addresses, phones
  WHERE addresses.name=phones.name
```

In this example the mediated view completebook is composed of the union of addressbook with the join of addresses and phones.

A drawback of Global as View is that extending the system with additional information sources is hard, since the mediated schema has to be redesigned and rebuilt.

Local as view

Local as View considers the original information source to be derived from the mediated schema. It describes the information sources in terms of queries to this mediated information source. Given the previous completebook example, three views would be created, for addressbook, addresses and phones.

```
CREATE SOURCE addressbook AS
  SELECT * FROM completebook

CREATE SOURCE addresses AS
  SELECT name, room FROM completebook
```

and

```
CREATE SOURCE phones AS
  SELECT name, phone FROM completebook
```

The *information source* completebook does not exist, but is a virtual information source composed of actual information sources addressbook, addresses and phones. Adding an information source in this case, is easy as the information sources are independent of each other. Adding one source would result in adding one extra query.

Querying GAV or LAV

When querying a mediated schema using GAV, the conversion to queries on the underlying data sources is fairly straightforward. With SQL this can be accomplished by creating a view, while using XQuery, this can be achieved by executing the original query onto the XQueries that perform the transformation to the mediated schema.

In a system using LAV, queries are not easily transformed, because the mediated schema is not the view. Reconstructing which elements come from which data sources requires going through all the view definitions of the underlying data sources.

5.4.4 Schema matching

Before schemas can be integrated, they have to be matched first. In [RB01] an overview is given of approaches to automatic schema matching. A hierarchical classification of schema matching approaches is presented. Schemas can be matched based on a number of characteristics. Names of attributes or elements can be compared. If they match, or are relatively similar, the attributes are matched and the associated data can be integrated. Instance data can also be compared. If the data contained in an attribute is similar, it is likely that the attributes have to be merged. These methods base their decision on one property of one attribute and are therefore referred to as *element matching*. Other approaches use the structure of the schema itself, or a combination of attributes. These methods are called *structure matching*.

When trying to match two schemas R_1 and R_2, a number of situations can occur [BLN86]. The schemas can be

1. *Identical.* In this case R_1 and R_2 are exactly the same. This is the case for systems for which the same modeling constructs are used and the same perceptions are applied.

2. *Equivalent.* In this case R_1 and R_2 are not identical. This is caused by use of different modeling constructs. However, the perceptions applied

are the same and must be coherent. Definitions of equivalence are usually based on three different types:

(a) *Behavioural.* Two schemas are equivalent if for every instance of one schema, there is a corresponding instance of the second schema that gives the same answers to every possible query.

(b) *Mapping.* This is the case, when instances of both schemas can be put in a one to one correspondence, see [Ris77].

(c) *Transformational.* This types of equivalence holds, if one of the schemas can be transformed into the other by using atomic transformations that make the schemas behavioural or mapping equivalent.

3. *Compatible.* The two schemas are neither identical, nor equivalent, but they are also not in contradiction. This means that modeling constructs, designer perception and integrity constraints cannot be contradictory.

4. *Incompatible.* If two schemas are not identical, equivalent or compatible, they are incompatible. The two schemas contradict, because of the incoherence of the specification.

5.4.5 Learners

In the Learning Source Description (LSD) system [DDH03], learning modules, called *base learners*, are used. Each base learner uses well one certain type of information to find semantical mappings between schemas.

There are many different base learners. Examples include: Name Learners, Naive Bayes Learner, Content Learner and XML Learner. The Name Learner matches elements based on their name. In case of XML, this name is the tag name. The learner can use lists of synonyms to match similar, but not equal names. The Content Learner matches elements based on the content of the attribute. In case of XML, this is the data value of the XML element. The learners use a set of recognizers, which are each capable of recognizing a certain kind of item, i.e. a zip code or a phone number. A schematic example of learner process is shown in Figure 5.4.

The base learners are first trained in a phase called the *training phase.* During this phase, the user has to confirm the results so the learner *knows* if the results are correct. After this training phase, the learner can be used, this is called the *matching phase* [Hal04].

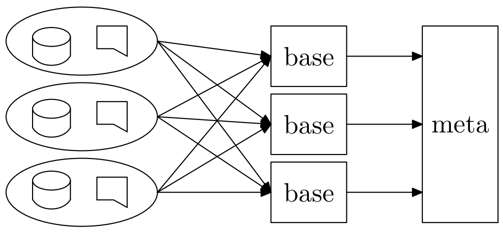


Figure 5.4: Using Learners

Base learners know how to map on schemas, using one technique. Meta Learners combine the results of base learners to match using multiple techniques. See [DDH03].

5.4.6 Using time

If we allow the schema of an information source to change over time, data stored at different moments can have different schemas. A query to this information source should, at a minimum (evolution), be possible using the latest version of the schema. To keep track of the version used for particular parts of the data, a notion of time can be added to the schema and data. There is much overlap with this kind of time usage and time in temporal databases.

Time can be used to help the schema integration process. When decisions about validaty need to be made, often newer schemas are considered to be more accurate than older schemas. In case of evolution and versioning, the order of schemas can be determined by looking at time of instances contained in the database.

Different notions of time can be used to accommodate for different functionality of the system. Keeping track of the original schema of a particular part of the data is one aspect, but also rolling back the information source to a prior state, or being able to answer time related queries, are possibilities of using time.

Valid time

The first notion of time that can be included in an information source is the time at which the recorded event occurred. Suppose we change the address of one of the persons in our address book, indicating this person has moved. We can attach a time stamp to this record indicating from which point in time this record is valid. This kind of time is referred to as *valid time*.

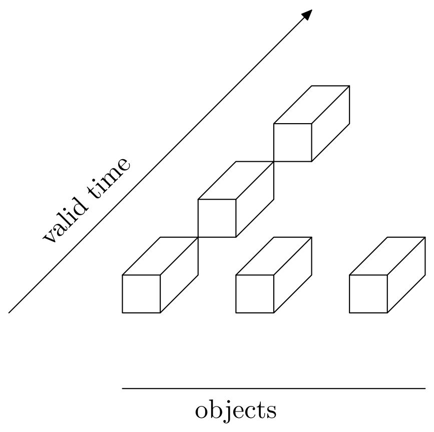


Figure 5.5: Valid Time

With valid time, we can ask the information source questions like: *What was the address of person X on February* 2^{nd} *last year.* A schematic representation of valid time is given in Figure 5.5

Transaction time

Another notion of time, is the time when the record was inserted into the information source. If all the records are associated with such a time stamp and provided that deletions are also recorded, we can reconstruct the information source for a certain date and time. The time recorded is the time at which the transaction occurred that created that particular record, and is therefore referred to as *transaction time.* Transaction time enables the system to do a roll back to any particular point in time. This is shown in Figure 5.6.

User defined time

The last notion of time we discuss, is already present in current relational database systems. If the user wants to represent time, he can use the DATETIME type. The database system itself regards this information as just another type of data. From the database point of view there is no special meaning associated with this type of time and it is therefore known as *user defined time.*

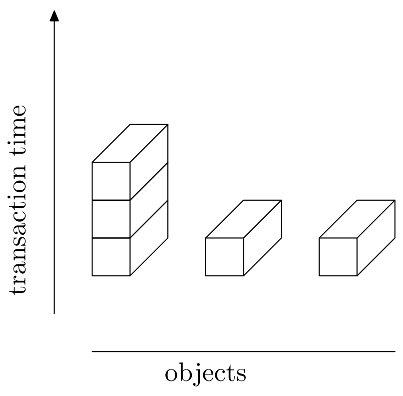


Figure 5.6: Transaction Time

5.4.7 Semantics of schema

As indicated earlier, the semantics of schemas is important when integrating data sources. Only semantically similar data can meaningfully be integrated. A possible and commonly used solution when the semantics of schemas is unclear, is to let the user provide the semantics. Most of the time, this is done by letting the user specify how the data has to be integrated. In the case of the LSD system, this is achieved by first specifying mappings manually in the training phase. Next, in the matching phase, the system can match the schemas automatically.

Another method to collect semantical information is to use an ontology, indicated by [Ver97]. This process, however, is error prone since ontologies may be outdated when used in a new setting. Therefore, when using ontologies they should be up to date and accurate. Another problem that arises when using ontologies for integration of information sources, is that they would have to support versioning and/or evolution as well [NK]. This is needed, because when schemas evolve, the matching concepts in ontologies should be able to evolve as well.

5.5 Data Integration

From this point on, we assume schema integration to be resolved and we will focus on the problem of data integration. At data integration time, elements from different sources have to be merged into a new data source. There can be, and usually is, overlap between data from one source and data from the other source. By overlap we mean elements referring to the same real world object, but not necessarily containing exactly the same information, or even the same kind of information. This was already shown in Table 5.1 where the two tables contain address books, but only the first table (Table 5.1(a)) also contains the email address.

Because the data sources may not contain the same kind of information, it is difficult to make a positive decision about equality of elements. Partly because of the semantics hidden in the schema. If, for example, the name elements don't correspond the likelihood of address book elements referencing the same person is low, whereas if just the email addresses don't correspond and taking into account that people have more email addresses and even change their address from time to time, the likelihood here is higher.

5.5.1 General approach

A device's database is a probabilistic XML document. When data integration with a foreign probabilistic XML document is initiated, the foreign document is considered to be a source of 'new' information on real world objects the device either already knows about or not. New information on 'new' real world objects is simply added to the database. Any differences in information on 'existing' real world objects are regarded as different possibilities for that object. Note that we disregard possibilities concerning order. New information on 'new' real world objects is simply considered to come after information on known objects in document order.

Since it is often not possible to determine with certainty that two specific XML elements correspond to the same real world object, we use a *rule engine* that determines the probability of two elements referring to the same real world object. In special cases, this rule engine may obviously decide on a probability of 0 (with certainty *not* the same real world object) or 1 (with certainty the same real world object). The rule engine, or Oracle, will be explained in section 5.6. In this section, we abstract from the details of the Oracle, but imagine that it uses schema information to rule out possibilities. Or it may, for example, consult a digital street map to declare a certain street name very improbable as there exists no such street in that city. Or it may use Semantic Web techniques to reason away possibilities.

On top of the Oracle, the integration system uses two rules to further limit the decisions about integration.

- The schema states that a certain element can appear only once. We assume that this means that the elements of both documents refer to the same real world object, hence, the subtrees are correspondingly merged. For example, if two person elements refer to the same real world person, their descendant elements that are declared in the schema as appearing only once (e.g., nm and tel) are merged. If two corresponding descendant elements differ, we store this as two possibilities for that element.

- The schema states that a certain element can appear multiple times. We assume that this means that the foreign document may contain new elements for this list. For example, the database contains knowledge about two persons "John" and "Rita". The foreign document holds information on a person "Jon". Note that "Jon" may be the same person as "John" only misspelled, or it may refer to a different person. The data integrator will store both possibilities, i.e., one whereby it merges "John" and "Jon", and one whereby it adds a new person element.

Each possibility is assigned a probability by the Oracle. For example, it is not unthinkable that "Jon" and "John" are actually the same person. On the other hand, it is rather improbable that "Rita" and "Jon" are the same person.

The minimal set of rules used by our prototype also includes that there can only be one root in an XML document and schema's of integrated documents are the same, so different tag names are assumed to refer to different real world objects.

5.5.2 Integrating sequences

In general, integrating sequences produces possibilities for all elements referring to either the same or different real world objects. Since we made an assumption that the schemas are the same and that elements with different tag names refer to different real world objects, many of those possibilities are ruled out. However, this rule does not limit the possibilities for sequences of elements with the same tag name.

Example 8 *We integrate address information of people. We are confronted with integrating sequences of person elements. Because our basic Oracle contains very limited knowledge, any two elements, one from each sequence, possibly refer to the same real world object. Therefore, when merging two sequences, X and Y, the resulting number of possibilities can be huge.*

Let, for example, $X = [A, B]$ and $Y = [C, D]$. The possibilities to be generated during integration of X and Y are listed in Table 5.2. In the table, $A = C$ indicates that A and C are considered to refer to the same real world object, hence, they should result in a single possibility where A and B are merged: A/B. Since the database already represents all possibilities explicitly, we do not need to consider two elements from one sequence to refer to the same real world object, so $A = B$ and $C = D$ are not valid possibilities.

Referral to real world object	resulting sequence
$A \neq B \neq C \neq D$	A, B, C, D
$A = C, B \neq C \neq D$	$A/C, B, D$
$A = D, B \neq C \neq D$	$A/D, B, C$
$A \neq C \neq D, B = C$	$A, B/C, D$
$A \neq C \neq D, B = D$	$A, B/D, C$
$A = C, B = D$	$A/C, B/D$
$A = D, B = C$	$A/D, B/C$

Table 5.2: Possibilities for merging sequences $x = \{A, B\}$ and $y = \{C, D\}$

Integration formally defined

We now formally define integration of two sequences of elements. There are two sets that are important during this phase. The first set is those element combinations that are certainly referring to the same real world object. This subset is called $Must$ and contains exactly those (element, element) combinations for which the Oracle predicted a confidence score of 1. The other important set during this phase, is called Not and contains all those (element, element) combinations for which the Oracle predicted a confidence score 0 and these combinations should therefore not be included in the integrated document.

We first introduce some notation.

$$A \rightarrow B = \{f \subseteq A \times B | (\forall a \exists_1 b \bullet (a, b) \in f\}$$
$$A \leftrightarrow B = \{f : A \rightarrow B | (\forall a, a' \bullet fa = fa' \Rightarrow a = a')\}$$

Given $A, B, Must : A \leftrightarrow B, Not \subseteq A \times B$
The integrated document R is defined as follows:

$$R = \{f : A \leftrightarrow B | Must \subseteq f \wedge (f \cap Not = \emptyset\}$$

Document R contains all those sets of (element, element) combinations, such that it includes $Must$ and does not include any of the (element, element) combinations of Not.
We now define a function $Compl$ that completes the integrated document with all possibly integrated elements. We start with $Must$ and add to it those (element, element) combinations that are not in Not. The result of $Compl$ is the fully integrated, uncertain document.

$$Compl(f, A', B') = \{g : A' \leftrightarrow B' | g \cap Not = \emptyset \bullet f \cup g\}$$

then,

$$Compl(Must, A \setminus dom(Must), B \setminus ran(Must)) \quad (5.1)$$
$$= \{g : (A \setminus dom(Must)) \leftrightarrow (B \setminus ran(Must)) | g \cap Not = \emptyset \bullet Must \cup g\} \quad (5.2)$$
$$= \{f : A \leftrightarrow B | f \cap Not = \emptyset \wedge f \supseteq Must \bullet f\} \quad (5.3)$$

From step 2 to 3 we assume $Must \cap Not = \emptyset$. This assumption always holds, since $Must$ contains exactly those element combinations where the Oracle predicted a confidence score of exactly 1, whereas Not contains all those combinations where the Oracle returned a confidence score of 0. The Oracle only provides one confidence score for each (element, element) combination and can therefore never return both 1 and 0 as scores. As a result $Must \cap Not = \emptyset$ always holds.

As can be seen from the formal definition, the number of possibilities generated from integrating two sequences is large, even for small documents. We show how many possibilities are generated if no world knowledge is taken into account. When all elements of X and Y refer to other real world objects, the number of resulting possible worlds is 1. But, when one element from X refers to the same real world object as an element from Y, there are $X \times Y$ possible ways how this can be done, since every element from X can in principle be matched with every element from Y.

In general, if i elements from X match with i elements from Y, then the number of possible ways to merge i elements from X with i elements from Y can be computed as follows. In the following, $i < min(x, y)$, where x is the number of elements in X and y is the number of elements in Y.

Choose i different elements from X, where the order of choosing the elements is unimportant, but an element cannot be chosen more than once. This can be done in $\binom{x}{i} = \frac{x!}{(x-i)!i!}$ ways. Then, we choose i elements from Y to merge with those chosen from X. Since the first chosen element from X should be merged with the first element chosen from Y, order is important when choosing elements from Y. The number of ways to choose the i elements from Y is $\frac{y!}{(y-i)!}$.

The process of merging sequences is commutative, we assume $x \leq y$. In determining all possibilities, any i $(0 \leq i \leq x)$ elements of X may refer to the same real world object as elements of Y. Therefore, the resulting total number of possibilities for a merged sequence is

$$\sum_{i=0}^{x} \binom{x}{i} \times \frac{y!}{(y-i)!}$$

We see from this formula, that merged documents can become huge quite rapidly. If we take, for example, $x = 5$ and $y = 5$, then the maximum number of possibilities is 1546. The rule engine, however, may rule out certain possibilities. For example, if in the case of Table 5.2, A refers to a person named "John" and C to a person named "Rita", the rule engine may assign probability 0 to the likelihood that $A = C$. In this way, it rules out two of the seven possibilities.

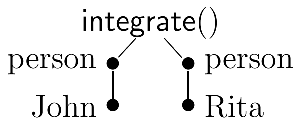
Integration method implementation

We implemented the integration method in XQuery. The implementation is kept as close to the formal definition as possible. A pseudo code implementation of the algorithm is given in Figure 5.7.

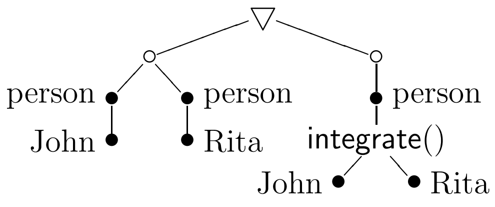
Below, we first show an example of integrating two certain trees to illustrate the recursive process.

The data integration function integrate takes two parameters D_1 and D_2. It returns the integration result as a probabilistic XML tree. In the diagrams below, we have omitted probability and possibility nodes whenever there is only one possibility.

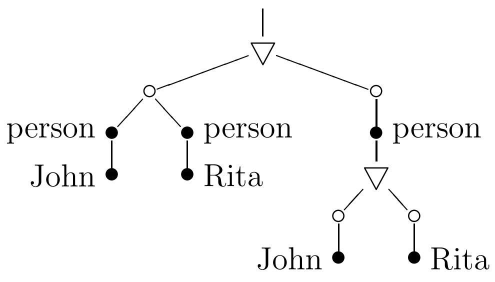
The example below shows how we can recursively integrate two certain trees.



After the first integration step, we obtain:



The second integration step integrate('John', 'Rita') results in the final integrated document



```
complete(A,B,Not) : $\mathbb{P}(A \times B)$
begin
    result := $\emptyset$
    for each $a \in A, b \in B$
        if ( $(a \mapsto b) \notin Not$ )
        then c := complete($A \setminus \{a\}$,$B \setminus \{b\}$,Not)
            result := result $\cup \{c, \{(a \mapsto b)\} \cup c\}$
    return result
end


combinations(A,B) : $\mathbb{P}(A \times B \cup A \cup B)$
begin
    mustbe := $\emptyset$; not := $\emptyset$; result := $\emptyset$
    for each $a \in A, b \in B$
        o := oracle(a,b)
        if (o = 1) then mustbe := mustbe $\cup (a \mapsto b)$
        else if (o = 0) then not := not $\cup (a \mapsto b)$
    c := complete($A \setminus$ dom(mustbe), $B \setminus$ ran(mustbe), not)
    for each f in c
        f' := f $\cup$ mustbe
        result := result $\cup$ {f' $\cup$ A \dom(f') $\cup$ B \ran(f')}
    return result
end


integrate(E1,E2)
begin
    if (E1 and E2 are text nodes)
    then if (E1/text() = E2/text())
        then result := <prob><poss>E1</poss></prob>
        else result := <prob><poss>E1</poss><poss>E2</poss></prob>
    else
        A:=E1/child::node(); B:=E2/child::node(); E:=E1/name()
        comb := combinations(A,B)
        result := <prob/>
        for each f in comb
            p := <poss/>
            for each m in f
                if ("m of the form $(a \mapsto b)$")
                then p.addchild(<E>integrate(a,b)</E>)
                else p.addchild(m)
            result.addchild(p)
    return result
end
```

Figure 5.7: Integration Algorithm

Observe the difference between integrating person elements, which are specified as being part of a sequence, and other elements for which there can only be one, for example the name node. The former produces an additional possibility for the case that there exist two persons. In general, text nodes are also part of a sequence (e.g., paragraphs in a text document). Concatenating names of persons, however, does not make sense, so the integration system decides that, for example, the name of a person can not be “JohnRita”.

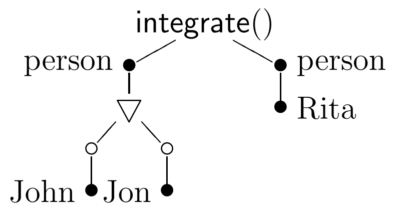
5.5.3 Equivalence preserving operation

An interesting property of the data integration approach described above, is that it preserves equivalence. Let D_1 and D_2 be two probabilistic XML documents and $A = \mathsf{integrate}(D_1, D_2)$ the result of integrating them. Suppose D'_1 and D'_2 are equivalent to D_1 and D_2 respectively. Is then $A' = \mathsf{integrate}(D'_1, D'_2)$ equivalent to A? Although proving equivalence preservation is feature research, we give an example that illustrates this property.

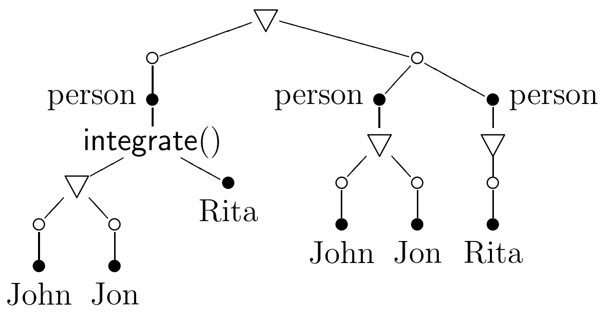
There is a special case for which this property is especially interesting. The set of possible worlds can be represented as a probabilistic tree with one probabilistic node as root and all possible worlds as possibilities directly below it. Figure 3.3(a) is of this form. Since the above property holds, integrating two probabilistic trees amounts to integrating all combinations of possible worlds of both trees.

We first show the integration of a compact tree with two possibilities with a certain tree. Next, we show the integration of an equivalent tree in set-of-possible-worlds representation with the same certain tree.

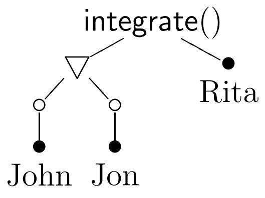
The algorithm presented in the previous section, integrates two certain documents, producing one uncertain integrated document. Here we show based on an example, the method to integrate a probabilistic tree and a certain document.



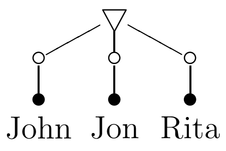
We would first integrate both person elements:



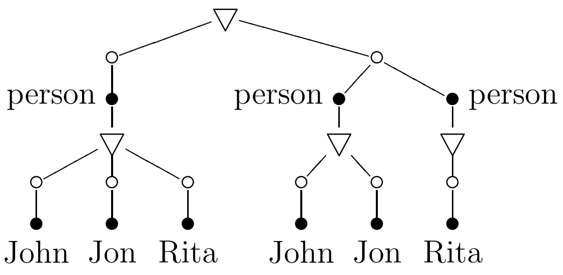
where



intuitively leads to



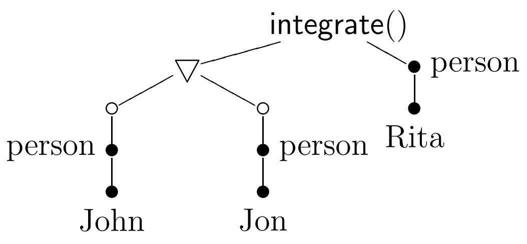
The entire resulting tree looks like:



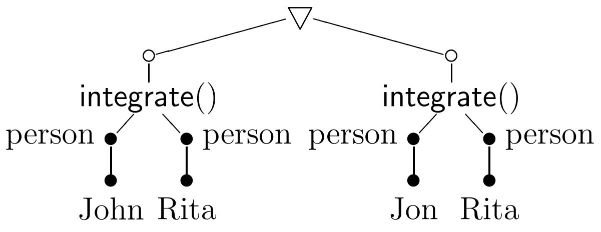
If we restrict ourselves to names of persons, the resulting document can be described using a simplified form of boolean notation, as:

$$(\text{Rita} \vee \text{John} \vee \text{Jon}) \vee ((\text{John} \vee \text{Jon}) \wedge \text{Rita}) \tag{5.4}$$

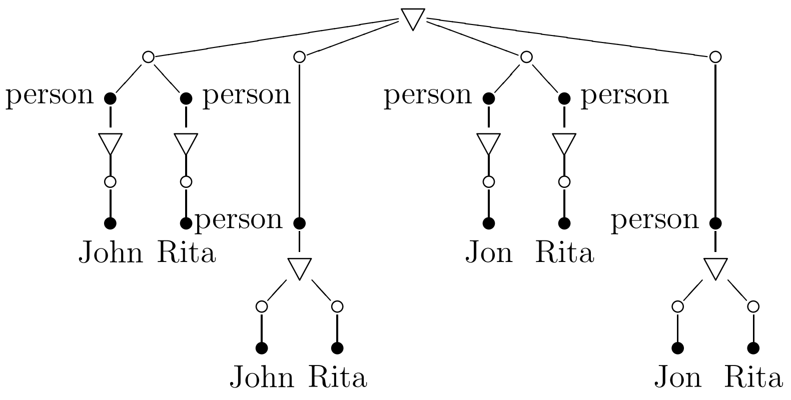
Moving the local possibility upwards in the tree, we get an equivalent less compact tree that is in all-possible-worlds representation. The integrate function now behaves as being applied to each possible world separately.



We integrate the person named "Rita" over both possibilities resulting in the following:



The final result is:



The boolean representation is

$$((\text{John} \wedge \text{Rita}) \vee (\text{John} \vee \text{Rita})) \vee ((\text{Jon} \wedge \text{Rita}) \vee (\text{Jon} \vee \text{Rita})) \qquad (5.5)$$

Note that this is equivalent to the earlier obtained boolean representation. The trees are equivalent.

5.6 The Oracle

As explained earlier, the rule engine determines in some way the probability of the various possibilities. Reasoning and the use of information from schema and other data sources can be used to limit the number of possibilities, but also to better assign probabilities.

There are several simple schemes to assign probabilities. A first scheme, which can be described as the *doubter*, would assign probabilities as follows. Whenever the database conflicts with a foreign document one some element, we assign probability .5 to each resulting possibility. This approach has a severe drawback. For example, when a mobile device has once heard from another device that a person's name is "John" and it meets another device which says its name is "Jon", the scheme assigns the possibilities "John" and "Jon" each a probability of .5. But, when it has already heard from ninety-nine different devices, that the name of the person is "John", it should be very suspicious when it meets a device that says this person's name is "Jon". Therefore, it should give the possibility "Jon" a very small probability of, say, .01.

The basic rule engine of our prototype has a still simple but sufficient scheme for assigning probabilities. It is based on the premise that what you have seen twice, is twice as likely to be correct. In other words, a *confidence score* should be kept in the data. This factor is an indication of how certain we are about the data, which helps in assigning probabilities when integrating new data. Every time a different device claims a certain possibility is true, the confidence score is increased by one.

In addition, to decide if two elements refer to the same real world object, we introduced a new component called The Oracle. The Oracle compares two elements and returns the likelihood that these elements are equal.

5.6.1 Entity Resolution

Entity resolution is the process of identifying elements that refer to the same real world object. With probabilistic data, the result of entity resolution is an uncertain answer with an associated probability.

Definition 8 *A match function is a function of the form* $m : (n_1, n_2) \mapsto [0, 1]$*, that indicates to what extent elements* n_1 *and* n_2 *refer to the same real world object, using the criteria of that particular match function.*

Match functions can be designed to work on different levels in the XML tree. One match function, for example, can be used to specifically match phone numbers, while another match function can be used to match complete person elements. Therefore, a combination of different match functions needs to be used to obtain a correct result. Combining the results of different match functions, is the task of a component called The Oracle.

Definition 9 *The Oracle is a function of the form $O : (n_1, n_2) \mapsto [0, 1]$. The result of this function is the weighted average of match functions.*

The result of different evaluation functions are combined using a weighted average and returned as the result of the Oracle.

Example 9 *Consider the movie example, where movies have a title, a year and a set of cast members. Match functions can be defined for all of these elements and an additional match function can be defined for movies as a whole.*

5.7 Summary

The uncertainty model described in chapter 3 is used in the integration method described in this chapter. By allowing the result of the integration application to be uncertain, the results can directly be used without involvement of a human to make decisions on equality.

Chapter 6

Reducing Uncertainty

Although the goal of uncertain data management is to be able to capture possible uncertainty in the data, it is desirable to limit the amount of uncertainty. In all applications discussed previously, except for updating uncertain data, the amount of uncertainty can only grow. In this chapter, we introduce two methods to reduce the amount of uncertainty in the data. The first method, using knowledge rules during integration time, doesn't require any user interaction, while the second method, user feedback, requires user interaction at query time. First, we'll present a movie database scenario where two movie databases are integrated. This example shows the explosion of uncertainty and is used to illustrate both reduction methods.

6.1 Movie database scenario

Without world knowledge, integrated information sources can become very large. As explained, this is due to the fact that many things, however remotely possible, are *in principle* possible. The example of Section 5.5.2 shows that for the integration of two data sources with each two data items, there are already seven possible worlds. In [KKA05], we calculated that for two data sources with each five data items each carrying four children, there are in theory 1546 possible combinations between the elements from both information sources.

But this is theory and these are small examples. To be able to get a feel for the size of the problem *in practice*, we investigate a scenario in which we attempt to integrate a number of data sources on the web containing movie information. Table 6.1 shows several interesting data sources that may be used. There are many movie data sources with similar kinds of information about many movies, such as IMDb, All Movie Guide and Yahoo! movies. But

Source	**#movies**	**Description of information offered**
Internet Movie Database [IMD]	470,000	movie details, plot summary, cast, other people involved, genre, goofs, quotes, trivia, user comments, awards, user rating, etc.
All Movie Guide [AMG]	290,000	movie details, plot summary, cast, other people involved, genre, keywords, themes, moods, etc.
Yahoo! movies [Yah]	*unknown*	movie details, plot summary, cast, other people involved, genre, user comments, photos, critics reviews, etc.
Simply Scripts [SS]	1,500	title, directors, transcript

Table 6.1: Some movie sources

there are also data sources that have very specific data about only a limited number of movies, such as Simply Scripts offering transcripts of what is said in a movie. Integrating the information of such data sources may have much added value.

Main cause semantic equality problem

The main cause for explosion in the number of possible worlds is the *semantic equality problem*: How to decide whether or not two data items refer to the same real-world object? Without world knowledge, any movie data item may in theory be semantically equal to any other movie data item in another data source. This is caused by the fact that without world knowledge a system is unable to decide where the boundary is between two descriptions referring to the same real world object or not. When integrating sources with hundreds of thousands of movies, the number of possibilities are enormous, most importantly too many to handle. In the following example, we show the semantic equality problem.

Example 10 *Consider the two XML documents from Figure 6.1. In the first XML document (Figure 6.1) the address information about a person named Elisabeth Stone is stored. The second document contains information about a person Beth Cold. If we just consider the first names, we could conclude that both documents are about the same person. In the first document, her*

```
<person>
        <name>Elisabeth Stone</name>
        <tel>1234</tel>
        <address>610 El Camino Real</address>
        <city>Palo Alto</city>
</person>

<person>
        <name>Beth Cold</name>
        <tel>5678</tel>
        <address>2020 California Ave.</address>
        <city>Mountain View</city>
</person>
```

Figure 6.1: Semantic equality problem

official name, Elisabeth, is used, but she is normally called Beth, which is stored in the second document After the creation of the first document, she could have married a person with last name Cold and together have moved to a new address, also obtaining a new telephone number.

Some features of a data item, however, are often known to be keys or key-like, hence can be used to determine whether or not two data items refer to the same real-world object. For movies, we found for example many data sources including the IMDb number. This number can be used to determine with absolute certainty that two movie data items actually refer to the same movie. There is an important assumption here not to be neglected: This assumes that the data is correct. If not, we may erroneously decide that two data items are the same. If we drop this assumption, we need to consider the possibility that even if IMDb numbers are equal, this does not automatically determine equality on movies. Furthermore, it may also happen that two data items that do refer to the same movie, are not detected as such, resulting in duplication of information. Unfortunately, not all movie data sources include the IMDb number, but they do include the title, an attribute that is also almost always correct and very discriminative. But it is not a key: there are, for example, three movies called "King Kong", namely the 1933, 1976, and 2005 versions. Together with the year attribute, which is almost always also included, we do have a good alternative key.

Other causes for explosion in the number of possible worlds are differences in which attributes are included and actual attribute values that do not correspond. To get a feel for this problem, Table 6.2 investigates in more detail the data given for the 2005 "King kong" movie by the first three data

Attribute	**Comparison**
Title/Year	Exactly equal in all three sources.
Genre	IMDb gives 'Action', 'Adventure', 'Drama', 'Fantasy', 'Sci-Fi', and 'Thriller'. AMG gives 'Adventure', 'Monster Film', and 'Period Film', hence only one in common. AMG has other genre-like attributes like keywords, themes, tones, and moods, but these do not overlap with any IMDb genre. Yahoo gives 'Action/Adventure', 'Romance', 'Thriller', and 'Remake'.
Cast	IMDb presents a cast of 15 people, AMG 11, and Yahoo 13. All provide both the names of the actors as well as whom they play in the movie. The 11 actors of AMG are all present in both IMDb and Yahoo. The 2 extra actors of Yahoo are different from the 4 extra of IMDb. Furthermore, there are three differences in spelling.
Location	IMDb has a 'Country'-attribute with value "New Zealand / USA". AMG has 'Filming location' with value "New Zealand". Yahoo has 'Filming Locations' with value "Wellington, New Zealand (Campertown Studios - Stone Street Studios)".
Plot summary	All three sources have a different description or plot summary.

Table 6.2: Comparison of information on the 2005 movie "King Kong".

sources of Table 6.1. What we can observe is the following:

- The 'Title/Year' information can indeed be used to exactly match the corresponding items in all three sources.
- The 'Genre'-attribute contains more differences than correspondences. In general, other movies show more correspondences, but they almost never completely agree. This is due to the subjective nature of the attribute and the usage of different terms. Assuming different strings indeed represent different genres, our information integration approach will result in a list of terms for this attribute whereby for each term it is uncertain whether or not the term is actually a member of the list. 'Adventure' will be the only one about which certainty exists (provided that the integrator separates the combined "Action/Adventure" genre description of Yahoo).

- The 'Cast'-attribute is technically speaking also a simple list of strings. The difference with 'Genre' is that it is much more factual data, hence much less differences can be observed. Even with factual data, however, we observe that the three sources do not fully agree on 'the cast' of the movie. There are 11 names that belong to the cast with certainty, but there are also 6 more names that are given by only one of the three sources.

- The 'Location'-attribute also concerns factual data. Although all sources agree that the filming location is "New Zealand", the actual string values are far from the same. Integrating this attribute with the approach of Chapter 5 results in three possibilities for the attribute. The uncertainty, however, is local. A query asking for movies filmed in New Zealand contains a predicate like `contains(location,'New Zealand')` or even `location='New Zealand'`. In our probabilistic XML approach, such a query *will find the movie* "King Kong" although the query answer of the latter will have a lower probability assigned to this movie [KKA05].

- Attributes like 'Plot summary' completely differ for the three sources. In principle, all information sources are correct: All descriptions are valid descriptions of the movie. Similar to 'Location', our integration approach treats this as three local possibilities, which does not have significant negative effects on querying.

- Finally, if we were to also integrate Simply Scripts, we observe that it is possible to find the corresponding movies using Title/Year. Probabilistic XML trees can represent uncertainty about the existence of a subtree. Integration with Simply Scripts would result in local uncertainty about whether or not an attribute 'transcript' exists in the real world.

We can draw several conclusions from the analysis of the movie scenario. First of all, due to the existence of keys or key-like attributes, the explosion of possibilities resulting from the uncertainty about semantical equality of data items from different sources can be greatly reduced. The only uncertainty remaining is local for an attribute. With the compact representation of a probabilistic XML tree, the storage requirements for these local possibilities are not expected to be large [KKA05]. Furthermore, querying the resulting integrated data collection is not expected to suffer significantly from the incurred uncertainty. Items can still be found, some items may only have a reduced probability of being correct, because if the data is present before integration, i.e. at least one of the sources contains the correct information, it will also be included in the integrated document. Querying, in this case, returns the desired movie, possibly including different possibilities for some, or all of the subelements, but still containing all the desired information.

Most importantly, with some simple world knowledge statements such as 'Title/Year is a key', the number of possible worlds can be greatly reduced to a manageable size. Further world knowledge can even resolve local uncertainties, for example with statements like 'non-existence of a transcript is not a conflict, simply take it if available'. The same holds for genres and names in a cast.

6.2 Knowledge Rules

In Chapter 5, we did not use any world knowledge when integrating information sources. As a result, the number of possibilities in the resulting information source was huge. The size of this result can be reduced drastically, just by using very simple rules about the real world.

Knowledge rules can be either *generic*, such as

```
If two elements have at most one element for which the value
differs, the elements can possibly refer to the same real world
object
```

and *domain specific* rules, such as

```
If title elements of movies match, then the movies themselves
match
```

As can be seen from the above two examples, knowledge rules give an absolute statement about if two elements refer to the same real world object. According to such a rule two elements are either referring to the same real world object, or they are not. A knowledge rule can therefore be defined as a function that takes two elements as input and gives a boolean as output, indicating if the elements refer to the same real world object (true), or not (false).

Definition 10 *Let* $r : (element \times element) \rightarrow boolean$ *be an interface to a function that returns if, based on its implementation the two elements given as parameters possibly refer to the same real world object.*

In case the elements are considered not to refer to the same real world object, they are not integrated, hence not passed to the Oracle for evaluation. All element combinations that are positively evaluated by *all enabled* knowledge rules, are being passed to the Oracle. If none of the knowledge rules is enabled the Oracle determines if elements possibly refer to the same real world object alone. Knowledge rules, therefore are used to reduce the number of *possible matches*, instead of indicating if two elements actually refer to the same real world object. By using combinations of knowledge rules, the accuracy of the process increases.

Example 11 *In the movie example, we have defined several knowledge rules. The first knowledge rule is a refinement of the one given earlier, that states that two movies are not equal if their titles are not similar. Similar in this case is based on the edit-distance of the movies. Movies with titles "King Kong" and "Die Hard" would, according to this rule, not be considered to refer to the same real world object. As a result, they are not passed to the Oracle for comparison.*

Since in XML elements are nested, a knowledge rule can also use subelements of elements passed to the rule. In this way, more elements at the same time can be included in the decision process.

6.2.1 Experiments and Evaluation

In these experiments we show the effect of the rules we introduced. We show that some rules are more restrictive than others. In our experiments, we used documents with the following DTD.

```
<!DOCTYPE persons [
<!ELEMENT persons (person*) >
<!ELEMENT person (firstname, lastname, phone, room)>
<!ELEMENT firstname (#PCDATA) >
<!ELEMENT lastname (#PCDATA) >
<!ELEMENT phone (#PCDATA) >
<!ELEMENT room (#PCDATA) >
]>
```

We defined several knowledge rules. The first two are generic.

- Single Element Rule
  This rule considers object descriptions to refer to the same real world object, if *one or more* of the elements in both sources have the same value.

- 50% Rule
  This rule considers object descriptions to refer to the same real world object, if *at least 50%* of the elements in both sources have the same value.

Although the single element rule already rules out many possibilities, the 50% rule is much more restrictive and will eliminate more possibilities than the single element. Note that there is a trade-off between using more restrictive rules, resulting in smaller integrated documents, but possibly losing some desired matches and less restrictive rules that produce larger documents, but do produce these matches.

We also defined two domain specific knowledge rules, that take into account the fact that the data source contains information on persons, which have names.

- Firstname rule
  This rule considers data items to refer to the same real world object, if the firstname of both data items is equal.

- Lastname rule
  This rule considers data items to refer to the same real world object, if the lastname of both objects is equal.

We also defined some rules that are a combination of the previously defined rules.

- Combination rule 1
  This rule combines the 50% rule and the Firstname rule.

- Combination rule 2
  This rule combines the 50% rule and the Lastname rule.

- Combination rule 3
  This rule combines the Firstname rule and the Lastname rule and is therefore also referred to as the *Fullname rule.*

To keep the examples readable, we only show the name attribute of a person. Whenever the other elements have an effect on the integration process or result, we mention them specifically.

In our experiments we used the two address book documents given in Figure 6.2. Using the integration method without knowledge rules, the number of possible worlds in the result document is 1815. This explosion of possible worlds is caused by the fact that every element from the first source can possibly refer to the same real-world object as any element from the second source. For example, there is a remote possibility that even Mark Hamburg and Allen Kingship refer to the same person in reality.

We performed 7 experiments: one for every knowledge rule.

The simplest of the knowledge rules, the single element rule, already reduced the number of possible worlds to 39. This is a reduction of almost 98%, while the actual knowledge introduced is minimal: if two data items do not agree on any attribute, we decide that they do not refer to the same real-world object.

The 50% rule reduced the number of possible worlds to just 15, as do the firstname rule and combination rule 1.

The best result is achieved by using the Lastname rule, combination rule 1, or combination rule 3. These rules reduce the number of possible worlds to only 3. Combination rule 2 can be compared with combination rule 1, in the sense that a special emphasis is placed on one of the elements, in this case the lastname element.

We should, however, avoid adding world knowledge that does not hold in general. For example, if document 1 would have had the data item 'John Kingship / 4030 / 3035', it is actually very likely that this data item does *not* refer to the same real world object as 'Allen Kingship / 2020 / 3035'. The 50% rule is in this case not a good knowledge rule, because it rules out possibilities that are likely to be true. Good knowledge rules are those that have little or no false positives.

6.3 User Feedback

The previous section described the knowledge rules that can be used to reduce the amount of uncertainty. These knowledge rules are primarily used in The Oracle and therefore more suitable for applications such as data integration. The method described in this section is user feedback and is more application independent. We'll discuss the feedback process itself, the types of feedback that can be given and the effect feedback has on the uncertainty in the database in terms of the set of possible worlds. We will first extend the information cycle with the feedback process.

6.3.1 Information Cycle

When a query result is returned to the user, he is already involved with the system and feedback on the validity of the query result can easily be given. Uncertainty can be reduced by giving feedback on query results. Because the user posing the query also observes the real world, he can determine whether certain query answers are for certain correct or incorrect. By giving feedback in such cases, the database may conclude that certain possible worlds can no longer be correct and eliminate them. Feedback, in contrast with both integration and updating, can *never* introduce new worlds, or new elements in worlds.

The cycle of repeated observations and information integration introduces possible worlds, the cycle of repeated user feedback eliminates them. In this way, the uncertainty in the information in the database keeps reflecting the actual uncertainty about the state of affairs in the real world.

6.3.2 Types of Feedback

Consider the query given previously asking for the phone number of persons named "John". The answer (see Figure 6.4) is uncertain: either `("1111")`, `("2222")`, or `("1111","2222")`. A user could readily verify these answers, for example, by calling one or both phone numbers and checking if the person on the other end of the line is named "John". He could then indicate his findings by stating for some query results whether they are true or false in the real world. The goal of our user feedback technique is to use this information to update the information in the database accordingly, thus reducing uncertainty. We claim that a semantically correct way of doing this, is by invalidating entire possible worlds that disagree with the statement on the query result. For example, if a person named "John" picks up the phone when dialing "1111", then this is apparently a correct answer, hence any possible world not producing "1111" as an answer can be eliminated. This leaves two possibly correct possible worlds. Note that stating that "1111" is a correct answer, does not imply that "2222" in the answer is incorrect, since "2222" may be the phone number of another person named "John"; this corresponds with the third possible world.

We distinguish two types of feedback: *positive* and *negative* feedback. With negative feedback, the user indicates that one or more possibilities from the query result do not correspond with his knowledge of the real world. Positive feedback indicates that the user is certain that one or more possibilities from the query result correspond with the real world. Let RW_{user} be a user's *certain* knowledge of the real world. For simplicity, we represent RW_{user} with an XML tree, i.e. $\mathsf{RW}_{user} \in \mathcal{T}_{\text{fin}}$.

Definition 11 *Let $\overline{\mathcal{Q}}_q(PT)$ be a set of possible query answers for some query q and probabilistic XML tree PT, and $S \in \overline{\mathcal{Q}}_q(PT)$ In XQuery and XPath, a query answer is always a sequence, so we assume S to be a sequence.*

Negative feedback is a statement "a is false" for some $a \in S$. The meaning of this statement is $a \notin \overline{\mathcal{Q}}_q(\mathsf{RW}_{user})$. Analogously, positive feedback is a statement "a is true" meaning $a \in \mathcal{Q}_q(\mathsf{RW}_{user})$.

$\overline{\mathcal{Q}}_q(PT) = \{(\text{"1111"}), (\text{"2222"}), (\text{"1111"}, \text{"2222"})\}$ in our example, which in a system will probably be represented as $\{\text{"1111"}, \text{"2222"}\}$. The positive feedback that "1111" is a correct phone number means that "1111" $\in \mathcal{Q}_q(\mathsf{RW}_{user})$, i.e. the user states that the combination ("John", "1111') is for certain known by him. As a result, all worlds represented by the database, that do not contain ("John", "1111') are deleted from the database.

6.3.3 Effect of Feedback

As stated before, our approach is to invalidate, or rather eliminate, those possible worlds from the database that do not correspond with the user's knowledge of the real world.

Definitions 12 and 13 show how to construct the new possible worlds after giving positive and negative feedback, respectively.

Definition 12 *Let PT' be the result of user feedback "a is true" for some database PT, query q, and $a \in S$, where $S \in \overline{\mathcal{Q}}_q(PT)$. PT' is defined by* $\mathsf{PWS}_{PT'} = \{T \in \mathsf{PWS}_{PT} \mid a \in \mathcal{Q}_q(T)\}$

Definition 13 *Let PT' be the result of user feedback "a is false" for some database PT, query q, and $a \in S$, where $S \in \overline{\mathcal{Q}}_q(PT)$. PT' is defined by* $\mathsf{PWS}_{PT'} = \{T \in \mathsf{PWS}_{PT} \mid a \notin \mathcal{Q}_q(T)\}$

Observe that definitions 12 and 13 shows that we only need to eliminate possible worlds from the database. It is never necessary to create a possible world, create or delete a local possibility, or change or delete a part of a possible world. This is explained by the fact that from the set of original worlds, we select only those new worlds satisfying the feedback constraints, leaving out those that do not satisfy the feedback. As a result, whole possible worlds are either kept, or deleted. Because feedback deletes entire worlds, it is a powerful mechanism and should be used with caution. We will address this more thoroughly in section 6.3.6.

We have defined PT' by means of its possible worlds. Note that it is not hard to construct PT' from the set of possible worlds. Simply create a probability node with as many children as there are possible worlds. Attach each possible world, which is a certain probabilistic tree as subtree (see Figure 6.5). In this way, we obtain a probabilistic tree representing exactly this set of possible worlds. Any probabilistic tree equivalent with a PT' constructed in this way, preferably the compact representation, can be used as resulting database.

6.3.4 Recalculating Probabilities

When possible worlds are removed from the database as a result of feedback, the probabilities of all remaining possible worlds have to be recalculated. Unfortunately, the databases from which the probabilistic information source originated are typically unavailable. Therefore, re-integrating sources taking feedback into account is not a viable approach. Instead, we recalculate the new probabilities based on the probabilities that the remaining possible worlds had in the original database. Below, we argue that the correct way of recalculation amounts to simple normalization.

Our notation $\mathrm{P}(T \mid PT)$ suggests that we consider the database PT as the *universe*. To emphasize this fact, we use the symbol U for the original database. Eliminating possible worlds from this universe, means constructing a (new) database PT'. Let us first consider the case of a possible world T that is eliminated. Its probability $\mathrm{P}(T \mid PT')$ is, of course, 0. In the other case, we can calculate the probability of the possible world in the new universe using the laws of conditional probabilities as follows:

$$\mathrm{P}(T \mid PT') = \frac{\mathrm{P}(T \wedge PT')}{\mathrm{P}(PT')} = \frac{\mathrm{P}(PT' \mid T)\mathrm{P}(T)}{\mathrm{P}(PT')}$$

$\mathrm{P}(PT' \mid T) = 1$, because we are considering the case that T is a member of the universe, hence the existence of the new universe given possible world T is certain. The probability of the occurrence of the new database, i.e. the new set of possible worlds, is $\mathrm{P}(PT') = \sum_{T \in \mathsf{PWS}_{PT'}} \mathrm{P}(T)$. Note that $\mathrm{P}(T)$ is the probability of T *given our universe*, hence $\mathrm{P}(T) = \mathrm{P}(T \mid U)$. After substitution we finally derive

$$\mathrm{P}(T \mid PT') = \begin{cases} 0 & \text{if } T \text{ is eliminated} \\ \frac{\mathrm{P}(T \mid U)}{\sum_{T \in \mathsf{PWS}_{PT'}} \mathrm{P}(T \mid U)} & \text{otherwise} \end{cases}$$

As one can observe, the new probabilities can be obtained by simply normalizing probabilities. However, the calculation given above shows that normalizing probabilities semantically fits the possible world approach.

6.3.5 Properties of Feedback

For validation purposes, we analyze some desirable properties of our user feedback technique. This is a kind of analytical validation. For experimental validation, we refer to Section 6.4.

Property 1 *Given an original database PT and a resulting database PT' after user feedback, the amount of uncertainty does not grow, i.e.*

$$\mathsf{PWS}_{PT'} \subseteq \mathsf{PWS}_{PT}$$

This property follows directly from Definitions 12 and 13.

Property 2 *Given an original database PT and a resulting database PT' after user feedback, we observe that probabilities of possible worlds do not decrease*

$$\forall T \in \mathsf{PWS}_{PT'} \bullet \mathrm{P}(T \mid PT') \geq \mathrm{P}(T \mid PT)$$

This property follows from the formula derived in Section 6.3.4. $T \in \mathsf{PWS}_{PT'}$ means T is not eliminated. Since $\mathrm{P}(T \mid PT) = \mathrm{P}(T \mid U)$, we conclude that $\mathrm{P}(T \mid PT')$ is $\mathrm{P}(T \mid PT)$ divided by some number. Since $\left(\sum_{T \in \mathsf{PWS}_{PT}} \mathrm{P}(T \mid PT)\right) = 1$ and $\mathsf{PWS}_{PT'} \subseteq \mathsf{PWS}_{PT}$ (Property 1), this number is guaranteed to be larger than 0 and no larger than 1. Hence $\mathrm{P}(T \mid PT') \geq \mathrm{P}(T \mid PT)$.

Property 3 *The probabilities in the new database, $P(T|PT')$, are indeed a probabilistic distribution, i.e.*

$$\sum_{T \in \mathsf{PWS}_{PT'}} \mathrm{P}(T \mid PT') = 1$$

The property follows directly from substituting the formula from Section 6.3.4:

$$\begin{aligned}
&\sum_{T \in \mathsf{PWS}_{PT'}} \mathrm{P}(T \mid PT') = \\
&= \sum_{T \in \mathsf{PWS}_{PT'}} \frac{\mathrm{P}(T \mid U)}{\sum_{T \in \mathsf{PWS}_{PT'}} \mathrm{P}(T \mid U)} \\
&= \frac{\sum_{T \in \mathsf{PWS}_{PT'}} \mathrm{P}(T \mid U)}{\sum_{T \in \mathsf{PWS}_{PT'}} \mathrm{P}(T \mid U)} \\
&= 1
\end{aligned}$$

6.3.6 Give Feedback Carefully

We mentioned earlier that a database is a representation of the real world. Although this is true, there is a need for caution, because the real world changes, hence the observation of the real world can be different from the observation at a later time. Furthermore, knowledge about the real world is always incomplete. We denote the representation of the real world as captured in the database by RW. The representation of the real world as seen by the user at query time will be denoted by RW_{user}.

Due to the possible (non-)overlap between real world knowledge of the database and the user, feedback to a query in terms of absolute statements should be given with caution. We will show different scenarios of mismatch in knowledge and their impact on the feedback process.

Figure 6.6 shows four examples of observations from the real world contained by the database and the user. The examples are restricted to a set of names of people with the same name. In this case we show all people named "John". In each example, the left figure shows people named "John" *known* by the database and the right figure shows people named "John" known by the user.

Figures 6.6(a) and 6.6(b) show an ideal situation, where both the database and the user have knowledge about the same persons. Even though their respective knowledge of these persons may differ, there is no significant mismatch between database and user and the risk of wrong feedback is minimal.

Figures 6.6(c) and 6.6(d) show the situation where the knowledge of the database and that of the user is different, but the number of real world objects is equal. Here, the database has information on a person $John_2$ and the user doesn't, while the user has information on a person $John_3$ that is unknown to the database. In other words $\mathsf{RW} \neq \mathsf{RW}_{user}$. Feedback about the non-existence of John_2 by this particular user could result in the deletion of all possible worlds containing John_2, while in fact that person does exist, but is just not known to the user querying the database. The user should only give such negative feedback if he is certain that it is universal, i.e. that a database containing $John_2$ is for certain incorrect.

Figures 6.6(e) and 6.6(f) as well as Figures 6.6(g) and 6.6(h) show situations where the number of real world objects known by the database is also different than that known by the user. In such cases, feedback on queries with aggregates are likely to result in unwanted results and should only be given with special care. Suppose a user poses the query

```
let $grp := distinct-values(//person/name)
for $n in $grp
return
```

```
    <group> {
        $grp,
        count(//persons[./name eq $grp])
    }  </group>
```

to see how many people with the same name he knows, i.e. are contained in the database. It could happen that the query result for a name is different than he expects and he would like to give feedback on this.

For example, in the situation of Figures 6.6(g) and 6.6(h), the query result for “John” is 2, but the user knows 3 persons named “John”. Here the user should be aware that any feedback should not only be a universal truth, but also something the database with its incomplete knowledge should know. Giving the feedback that the query result should be 3 would eliminate all possible worlds with less than or more than 3 persons, hence one could possibly end up with an empty database.

Nevertheless, feedback can be a powerful mechanism in reducing uncertainty in the database if users (or application developers) use feedback with care, i.e. only universal truths or falsehoods, and only in cases where a database with incomplete knowledge should have knowledge about it. In other words, the database should have possessed the correct information.

6.4 Validation

Both knowledge rules and user feedback are validated in this section. We implemented a prototype that could integrate documents using knowledge rules and handle feedback from the user. We will discuss the prototype, the experiments we performed and the results of those experiments.

6.4.1 Prototype

For simplicity, we did not implement full querying capabilities, but only a function called *treeContainment*. This function checks if one XML tree, given as the first argument, is contained within another XML tree (second argument). Using *treeContainment* we can support both negative and positive feedback on simple path queries.

At feedback time, we extract all (probability, possible world) combinations from the current database and check whether or not each possible world meets the feedback criteria, i.e. whether the object is contained in the world in case of positive feedback, or whether the object is absent for negative feedback. The resulting database is represented by the set of possible worlds that meet the criteria for which the probabilities are normalized.

6.4.2 Experiments

In this section we will describe two experiments we performed using the prototype. In the experiments we used the integration result of the two address books shown in Figure 6.7. For readability, we sometimes use abbreviations for element names in our trees, e.g. ln for lastname. In the experiments we investigate the impact of both positive and negative feedback on the amount of uncertainty in the database. As a measurement of the amount of uncertainty, we count the number of possible worlds. Note that this number may exaggerate the amount of uncertainty as perceived by a human. For example, if a database contains three local uncertainties giving two possible values for three independent attributes, then there are *eight* possible worlds.

In the experiments, we query the information source and give feedback on the existence of a certain person, hence we don't make any statements about phone or room numbers. To be more precise, we execute the following query

```
for $person in //person
return <name>{p/fn," ",$p/ln}</name>
```

For positive feedback, we confirm the existence of the following 5 persons.

1. Mark Hamburg
2. Allen King
3. Stan Choice
4. John Friend
5. Allen Kingship

To confirm one name, we use one feedback statement. As a result the existence of all 5 people is confirmed after 5 positive feedback statements. Although we have confirmed the existence of 5 people, note that we only supported the fact that the name "Mark Hamburg" is used for a person in the real world. The number of people having this name can not be deduced from this feedback, nor the correctness of room or telephone numbers. Also, the fact that in reality "Allen King" and "Allen Kingship" may actually be referring to the same person, is no longer a possibility in one of the worlds after the feedback.

For negative feedback, we start again with the original document and indicate that any other name in the database is incorrect. Figure 6.8 shows the number of possible worlds at the start of the experiment and after each of the feedback statements, for both positive and negative feedback, but without using any of the knowledge rules. Figure 6.9 shows the same experiment, but now using knowledge rules at integration time to limit the number of possibilities in the integrated document.

A first observation is that Figures 6.8 and 6.9 confirm Property 1, because after every iteration the number of possible worlds in the information source decreases, or at least remains equal.

Experiment 1 In this first experiment we used no world knowledge during the integration process. The result is a huge document containing 1815 possible worlds. Each possible world has the same, albeit small, probability. Figure 6.8 shows the number of possible worlds after a number of iterations in the case of negative feedback and positive feedback.

The fact that positive feedback has more influence on the number of possible worlds than negative feedback can be explained as follows. At integration time, no world knowledge is used, therefore many possible worlds are created. Most of these possible worlds contain possibilities that do not reflect real world objects at all, and can perhaps best be described as "utter nonsense". In case of negative feedback, only those possible worlds would be removed, that exactly contain a person as in the feedback. With positive feedback, all possible worlds containing nonsense about the person subject to the feedback are eliminated, so only a few remain. In other words, many possible worlds are *dependent* on positive feedback, whereas only some possible worlds are dependent on negative feedback. The number of possible worlds remaining after 5 iterations of positive feedback is just 19. These possible worlds only contain persons really existing in the real world, only uncertainty about phone or room numbers remain.

Experiment 2 Probabilistic information integration without any world knowledge is not realistic in practice. Therefore, we repeat our experiment for an integrated information source for which we used one simple generic knowledge rule during integration to exclude the most nonsensical possibilities. The rule states that person elements can only refer to the same real world person if at least one of the attribute values is equal. The integrated database for this experiment contained 39 possible worlds.

In this case, positive feedback scores better than negative feedback as well. We can observe that with negative feedback, after 2 iterations, there is no improvement on the information source anymore. The remaining uncertainty is concerned with phone and room numbers. In case of positive feedback, only three possible worlds remain. Also in this case, the remaining possible worlds contain uncertainty that is associated with phone number and rooms and has to do with the fact that "Mark Hamburg" has two possible phone numbers, but it is also still possible that both instances of "Mark Hamburg" refer to different people. All other uncertainty is removed, because it is certain that which people exist (and therefore also which people don't exist).

For negative feedback we only indicated which people did not exist, but still for all remaining people many possible phone numbers and room numbers remain.

6.4.3 Results

In both experiments the number of possible worlds decreases and only those possible worlds containing false information (according to the feedback) are deleted. As a result, queries after feedback contain less false information.

The second experiment shows that using some very simple world knowledge at integration time, decreased the number of possible worlds drastically. With positive feedback on only the existence of people, without commenting on phone numbers or rooms, the number of possible worlds that remained was just 3.

The order of feedback for the end result, is not important. We could have given the feedback statements in any order, ending up with the same set of possible worlds. The reason is that all possible worlds containing or lacking (for negative and positive feedback respectively) *any* of the indicated answers, are removed from the set of possible worlds.

```
<persons>
<person>
<firstname>Mark</firstname>
<lastname>Hamburg</lastname>
<phone>1010</phone>
<room>3300</room>
</person>
<person>
<firstname>Allen</firstname>
<lastname>King</lastname>
<phone>2020</phone>
<room>3122</room>
</person>
<person>
<firstname>Stan</firstname>
<lastname>Choice</lastname>
<phone>3030</phone>
<room>3035</room>
</person>
<person>
<firstname>John</firstname>
<lastname>Friend</lastname>
<phone>4040</phone>
<room>3333</room>
</person>
</persons>
```

(a) Document 1 (660 bytes)

```
<persons>
<person>
<firstname>Mark</firstname>
<lastname>Hamburg</lastname>
<phone>1010</phone>
<room>3301</room>
</person>
<person>
<firstname>Allen</firstname>
<lastname>Kingship</lastname>
<phone>2020</phone>
<room>3035</room>
</person>
</persons>
```

(b) Document 2 (366 bytes)

Figure 6.2: Address book documents used in experiments

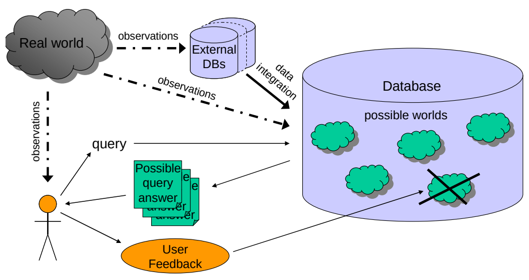
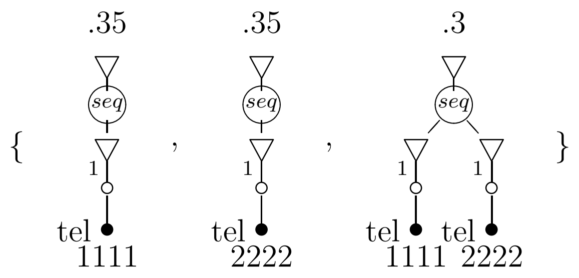
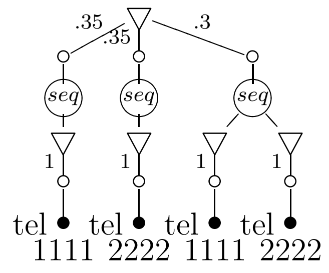


Figure 6.3: Information Cycle



(a) Set of possible query results



(b) Query result as probabilistic tree

Figure 6.4: Probabilistic query result

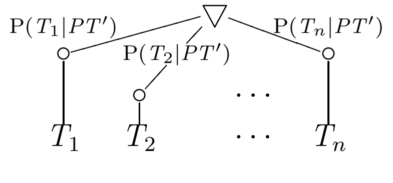
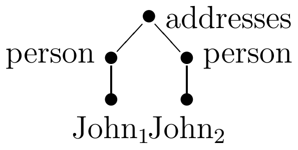
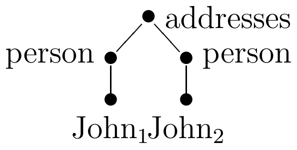


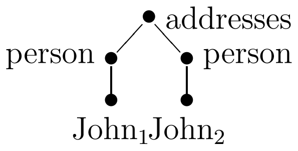
Figure 6.5: Construction of a probabilistic tree representation from its set of possible worlds



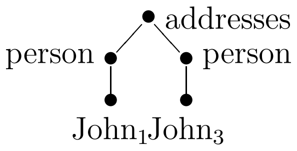
(a) DB1



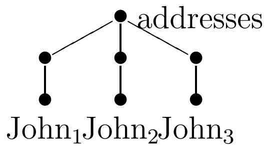
(b) Person1



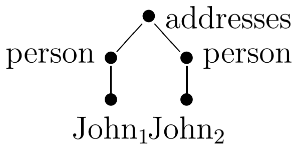
(c) DB2



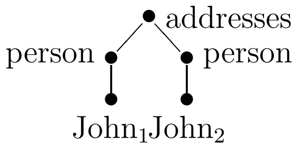
(d) Person2



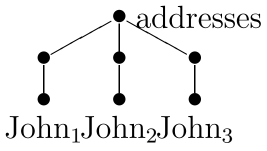
(e) DB3



(f) Person3

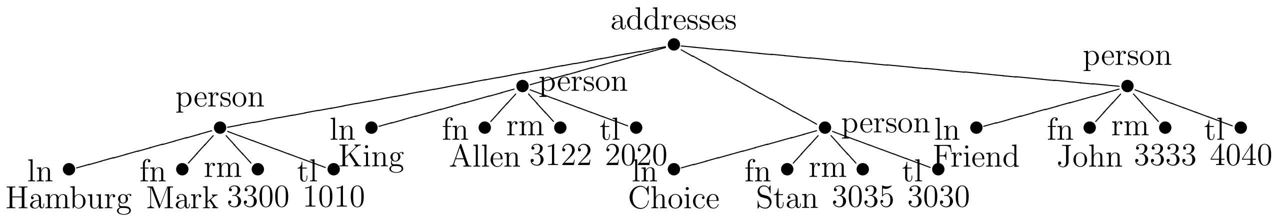


(g) DB4

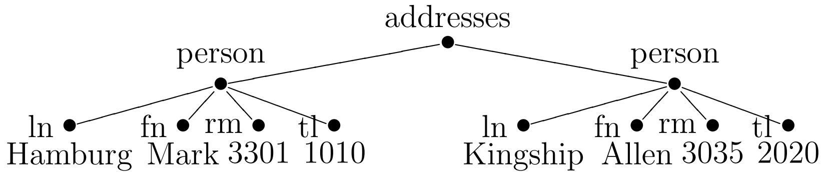


(h) Person4

Figure 6.6: Possible scenarios of DB - User (mis)match



(a) Address book 1



(b) Address book 2

Figure 6.7: Original address books

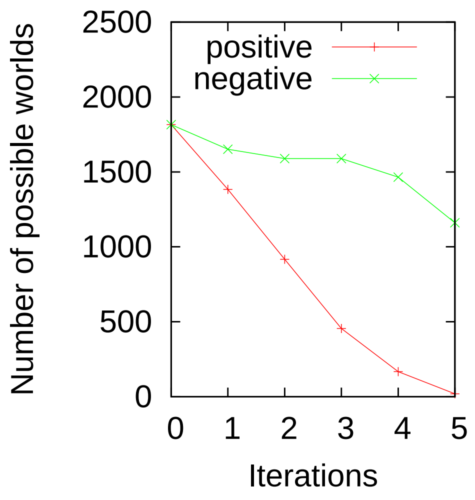


Figure 6.8: Feedback without world knowledge

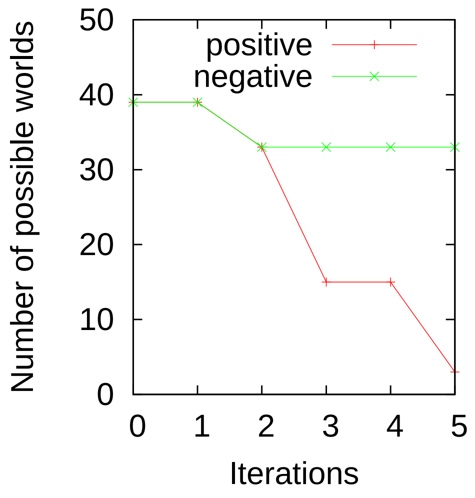


Figure 6.9: Feedback with world knowledge

Chapter 7

Conclusions

7.1 Summary

In this thesis, we introduced an uncertainty model based on XML with probability theory as a basis to calculate and propagate confidence scores. We introduced two measures to measure the amount of uncertainty present in the data and to be able to compare systems. We defined the semantics of queries on probabilistic XML in terms of possible worlds and showed how queries are evaluated using this semantics. We also modified precision and recall to be able to measure the quality of query results.

Using the probabilistic XML model, we showed that unattended integration at the data level is theoretically possible and results in correctly integrated documents. The document resulting from this integration method may contain many possible worlds. This large amount of possible worlds is a risk when it comes to scalability of the integration solution. Two methods are therefore introduced to reduce the amount of uncertainty. One method is specifically geared towards the integration process and involved introducing knowledge rules that eliminate possibilities during the integration process. Another method to reduce the amount of uncertainty is by allowing the user to provide feedback on query results. By indicating if a result is also the answer to the query in the real world, possible worlds can be eliminated. Either by specifying that the result is not an answer, and thus deleting all possible worlds that contain this result, or reinforcing the fact that a result is the answer to the specified query, thereby deleting all possible worlds that did not return this result.

7.2 Uncertainty Model

The uncertainty model introduced in this thesis supports

- probabilistic uncertainty
- mutual exclusiveness between elements
- dependency between elements
- sound semantics, being the possible world semantics

and is, as a result, an expressive means to capture, manipulate and query uncertain data. This is, in part, caused by using XML as an underlying data model. Since XML can be seen as a tree structure[1] the XML model itself already closely resembles a decision tree.

By storing probabilities locally, the *choice points* are kept as low in the tree as possible, which gives a compact representation of the possible worlds. By implementing the Directed Acyclic Graph structure proposed in chapter 3, replication of values and subtrees can be avoided further compacting the representation.

From the current systems known to us, none have either the expressiveness that is offered with our model, nor the compact representation that is used.

To quantify the amount of uncertainty in a document and to be able to compare between systems, we introduced two measures. The first measure, uncertainty density, quantifies the amount of uncertainty in the data. The second measure, answer decisiveness, quantifies the distinctiveness of the most likely possibility, averaged over the entire document. We also adapted precision and recall to be able to measure the quality of query answers.

7.3 Information Integration

In the area of information integration, we proposed to use uncertain data during the integration process in order to make this process autonomous. Theoretically, the integration process itself does indeed not need human involvement. In case of any doubt about whether or not two elements refer to the same real world object, the decision is postponed to query time. To capture the uncertainty about the integration two sets of possible worlds are dinstinguished, one where the elements are assumed to refer to the same real world object and one where do they not.

[1]At least, when ID/IDREF is ignored

By postponing the decision to query time, there is no need for human involvement at integration time. Instead a human (or application) can give feedback at query time. As a result, the integrated document can immediately be meaningfully used. Another benefit of this method is that the elements that are never in any query results will not need to be assessed by a human. Therefore, the total amount of time is shorter than in the traditional case and since the time needed is distributed over the queries, it is likely that the burden of integration, as perceived by the end user, is much lower.

7.4 Scalability

In practice, the number of possibilities grows exponentially. This leads to scalability problems. Although we have made some progress in this area by using knowledge rules to rule out possibilities, a more intelligent Oracle is needed to fully overcome the scalability problem. Also, using the DAG encoding presented in chapter 3 will help in reducing the size of the document, although not the number of possibilities. Querying using the possible world representation is also a drawback in terms of scalability. We are currently exploring possibilities to directly query the compact or DAG representation and although some progress has been made in the direction, we still have work in this direction before any conclusions can be drawn. We do expect that scalability, given a smarter Oracle, the DAG construct and directly querying the compact or DAG representation, can be kept within manageable limits in practice.

7.5 Research Questions

In this section we answer the research questions posed in chapter 1. The first question posed was

Which additions to existing data models are necessary to be able to support uncertain data resulting from information integration.

In chapter 3 we have shown a new data model for probabilistic XML that supports mutually exclusive possibilities for XML elements. In addition, we have a construct to support dependencies and independencies. Together, these constructs provide the mechanism to store probabilistic data that arises from information integration.

The second research question posed was

Which semantical foundation is needed to support intuitive querying on uncertain data.

We have presented the possible world approach and explained how, using this approach data stored using the probabilistic XML data model, can be queried (chapter 4). Also a mechanism to reduce the amount of uncertainty, namely user feedback, was presented that uses the possible world approach to eliminate possibilities (chapter 6).

To be able to compare systems, we posed the following question

How can uncertainty contained in documents and answer quality be measured.

In chapter 3 we presented two measures, uncertain density and answer decisiveness, to quantify the amount of uncertainty in the document and the ease with which most likely answers can be selected. In addition, we modified precision and recall to take into account the uncertainty associated with the answers as a measure for answer quality. Those four measures can be used to compare systems and query results.

The next research question addressed one of the possible applications using uncertain data, data integration.

How can uncertain database technology be theoretically applied in data integration.

We demonstrated the use of uncertainty in data integration in chapter 5. The integration method introduced there combines elements from source documents and when equality of the elements is uncertain, it stores this uncertainty in the integrated document. As a result, the document is integrated without involvement of the end user and decisions on equality are postponed, while the document can directly be queried.

As a result of introducing uncertainty in data integration, the size of the integrated documents becomes very large. The last question addresses that problem.

How can uncertain data in data integration be practically used in data integration.

To allow the user to reduce the number of possibilities in the integrated document, a feedback mechanism, presented in chapter 6, was introduced. Users can give both positive and negative feedback on query results, reducing the number of possible worlds represented by the database. We also introduced knowledge rules that reduce the initial number of possibilities and therefore the number of possible worlds in the integrated document.

7.6 Future Research

In this thesis, we have designed a probabilistic data model, based on XML and we have made a step in the direction of unattended information integration. Of course, the work does not end here and we have many directions for future work. We will list the most important ones below.

Uncertainty Model and Database

- Directed Acyclic Graph
  Although currently the theoretical groundwork for DAG constructions is done, we still need to work on querying this DAG structure directly. Without this work, querying is only possible when the DAG is converted to the possible world representation.

- Supporting ignorance
  Currently we do not take ignorance into account in the model. The solution chosen in Trio is that for missing probability mass, a questionmark is placed on the x-tuple. This option, we already support by adding an empty possibility node with the remaining probability. We envision to support a solution more along the lines of *coverage*, meaning that unspecified probability mass is used for uncovered element values. However, this requires a fundamental extension of the uncertainty model.

- Support for Continuous Uncertainty
  The probabilistic model used in IMPrECISE only supports discrete probabilistic distributions. Although some initial work has been done on extending this to continuous probabilistic distributions, there is no full theoretical foundation yet.

- Support for lineage
  Much like the Trio system, we plan to *keep track of where the data came from.* Especially in integration systems, this can be important metadata about the integrated document. If, at a later stage, data seems to be invalid, the original data can be tracked and recovered, even if feedback has been applied that states the original, correct data, was false.

Information Integration

- Increase in Oracle performance
  Our current Oracle performs slightly better than a truly doubtful Oracle, which would assign a uniform probability distribution to each of the child nodes of a probability node. We expect that, using knowledge from external information sources, like IMDb for movie data, or an online phone book, for address information, our Oracle can be improved significantly.

- Investigate the trade-off between a knowledgeable Oracle and relying on user feedback to come to a fully integrated document. A stricter Oracle will reduce the number of possibilities, but has a potential of dismissing correct integration options resulting in lower quality query answers, whereas a less restrictive Oracle will increase the size of the document and putting more effort on the end user giving feedback at query time.

- Combined Schema Integration and Data Integration mechanism
  Our current prototypes on schema integration and data integration, both work separately. It makes sense from a users perspective to combine the two, since in most cases integrating the documents completely, instead of just finding mappings, is the goal of this user anyway.

- Feedback on compact or DAG representation
  We have presented the theoretical foundation for a feedback mechanism in this thesis. Our current feedback mechanism, however, relies on the possible world representation. Giving feedback directly on the compact or DAG representation is currently an open question that will most likely be solved when querying on the compact or DAG representation is possible.

Bibliography

[AKO07a] Lyublena Antova, Christoph Koch, and Dan Olteanu. Maybms: Managing incomplete information with probabilistic world-set decompositions. In *ICDE*, pages 1479–1480. IEEE, 2007.

[AKO07b] Lyublena Antova, Christoph Koch, and Dan Olteanu. Query language support for incomplete information in the maybms system. In Christoph Koch, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Florescu, Chee Yong Chan, Venkatesh Ganti, Carl-Christian Kanne, Wolfgang Klas, and Erich J. Neuhold, editors, *VLDB*, pages 1422–1425. ACM, 2007.

[AMG] All movie guide. `http://www.allmovie.com`.

[AS06] S. Abiteboul and P. Senellart. Querying and updating probabilistic information in XML. In *Proceedings of EDBT, Munich, Germany*, pages 1059–1068, 2006. LNCS 3896.

[BDM+05] J. Boulos, N.N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *Proceedings of SIGMOD, Baltimore, Maryland, USA*, pages 891–893, 2005.

[BGMP90] D. Barbará, H. Garcia-Molina, and D. Porter. A probabilistic relational data model. In *Proceedings of EDBT, Venice, Italy*, volume 416 of *LNCS*, pages 60–74, 1990.

[BLN86] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[Bos07] S. Bosman. Map-it: An advanced multi-strategy and learning approach to schema matching. Master's thesis, University of Twente, April 2007.

[BP04] Patrick Bosc and Olivier Pivert. Possibilistic databases and generalized yes/no queries. In *DEXA Workshops*, pages 912–916. IEEE Computer Society, 2004.

[BSHW06] Omar Benjelloun, Anish Das Sarma, Chris Hayworth, and Jennifer Widom. An introduction to uldbs and the trio system. *IEEE Data Eng. Bull.*, 29(1):5–16, 2006.

[BYRN99] Riccardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999. ISBN 0-201-39829-X.

[CGMH$^+$94] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, 1994.

[CP05] R. Cheng and S. Prabhakar. *Encyclopedia of Database Technologies and Applications*, chapter Sensors, Uncertainty Models and Probabilistic Queries. Idea Group Publishing, 2005.

[CSP05] R. Cheng, S. Singh, and S. Prabhakar. U-DBMS: A database system for managing constantly-evolving data. In *Proceedings of VLDB, Trondheim, Norway*, pages 1271–1274, 2005.

[DDH01] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proc. SIGMOD Conf., Santa Barbara, USA*, pages 509–520, 2001.

[DDH03] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3):279–301, 2003.

[DH05] A. Doan and A. Halevy. Semantic integration research in the database community: Brief survey. *AI Magazine, Sp.Issue on Semantic Integration*, 2005.

[dKvK07] A. de Keijzer and M. van Keulen. Quality measures in uncertain data management. In H. Prade and V. S. Subrahmanian, editors, *Proceedings of the First International Conference on Scalable Uncertainty Management (SUM2007), Washington,* *DC, USA*, volume 4772 of *Lecture Notes in Computer Science*, pages 104–115, Berlin, October 2007. Springer Verlag.

[Doa02] A. Doan. *Learning to map between structured representations of Data*. PhD thesis, University of Washington, 2002.

[DS96] Debabrata Dey and Sumit Sarkar. A probabilistic relational model and algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.

[DS07] Nilesh N. Dalvi and Dan Suciu. The dichotomy of conjunctive queries on probabilistic structures. In Leonid Libkin, editor, *PODS*, pages 293–302. ACM, 2007.

[dV06] R. de Vos. The design and implementation of fleximatch: a learning, flexible & extendible framework for matching schemas. Master's thesis, Univ. of Twente, May 2006.

[GMHI$^+$95] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. rey, and U. Jennifer. Integrating and accessing heterogeneous information sources in tsimmis, 1995.

[GMPQ$^+$97] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[Hal04] Alon Halevy. XML and data integration, 2004.

[HGS03] E. Hung, L. Getoor, and V.S. Subrahmanian. PXML: A probabilistic semistructured data model and algebra. In *Proceedings of ICDE, Bangalore, India*, pages 467–478, 2003.

[IMD] The internet movie database (imdb). `http://www.imdb.com`.

[KKA05] M. van Keulen, A. de Keijzer, and W. Alink. A probabilistic xml approach to data integration. In *Proceedings of ICDE, Tokyo, Japan*, pages 459–470, 2005.

[LLRS97] Laks V. S. Lakshmanan, Nicola Leone, Robert Ross, and V. S. Subrahmanian. ProbView: a flexible probabilistic database system. *ACM Transactions on Database Systems*, 22(3):419–469, 1997.

[MTdK$^+$07] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. Das Sarma, R. Murthy, and T. Sugihara. Trio-One: Layering uncertainty and lineage on a conventional DBMS. In *Proceedings of CIDR, Monterey, USA*, pages 269–274, 2007.

[Nai03] Premchand S. Nair. *Uncertainty in Multi-Source Databases.* Springer Verlag, 2003.

[NK] Natalya F. Noy and Michel Klein. Systems ontology evolution: Not the same as schema evolution.

[PGMW95] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In P. S. Yu and A. L. P. Chen, editors, *11th Conference on Data Engineering*, pages 251–260, Taipei, Taiwan, 1995. IEEE Computer Society.

[RB01] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, Dec. 2001.

[RDS06] Christopher Re, Nilesh N. Dalvi, and Dan Suciu. Query evaluation on probabilistic databases. *IEEE Data Eng. Bull.*, 29(1):25–31, 2006.

[Ris77] Jorma Rissanen. Independent components of relations. *ACM Trans. Database Syst.*, 2(4):317–325, 1977.

[Rod95] J. Roddick. A survey of schema versioning issues for database systems, 1995.

[SL90] A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 1990.

[Smi06] M. Smiljanic. *XML schema matching: balancing efficiency and effectiveness by means of clustering.* PhD thesis, University of Twente, Zutphen, The Netherlands, April 2006.

[SS] Simply scripts, free movie scripts and screenplays. `http://www.simplyscripts.com`.

[Suc] Dan Suciu. Managing imprecisions with probabilistic databases slides. `http://www.cs.utwente.nl/~tdm`.

[Suc06] Dan Suciu. Managing imprecisions with probabilistic databases. In Ander de Keijzer and Maurice van Keulen, editors, *TDM*, volume WP06-01 of *CTIT Workshop Proceedings Series*, page 1. Centre for Telematics and Information Technology (CTIT), University of Twente, Enschede, The Netherlands, 2006.

[Ver97] M.W.W. Vermeer. *Semantic Interoperability For Legacy Databases*. PhD thesis, University of Twente, October 1997.

[Vis07] J. T. Visser. Finding nontrivial semantic matches between database schemas. Master's thesis, University of Twente, June 2007.

[Wid05] Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.

[Wij07] Jef Wijsen. Consistent joins under primary key constraints. In *Proc. of the first international VLDB workshop on Management of Uncertain Data*, pages 63–75, Sept. 2007.

[Yah] Yahoo! movies. `http://movies.yahoo.com`.

[Zad78] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1978.

[ZP97] E Zimanyi and A Pirotte. Imperfect information in relational databases. *Uncertainty Management in Information Systems, A. Motro and P. Smets, Eds.*, 1997.

Summary

In recent years, the need to support uncertain data has increased. Sensor applications, for example, are dealing with the inherent uncertainty about the readings of the sensors. Current database management systems are not equipped to deal with this uncertainty, other than as a user defined attribute. This forces the user of the DBMS to take on the responsibility of managing the uncertainty associated with the data.

In this thesis, we present a new data model, based on XML that is capable of storing uncertainty about elements and subtrees. The XML data model is extended in such a way, that probabilities can be associated with the elements and subtrees, dependency and independency of elements can be expressed and even the existence of entire elements or subtrees can be uncertain. We give a sound semantical foundation for dealing with the uncertainty associated with the data, and show how querying using this semantics works.

The probabilistic XML data model is used in an information integration application. Decisions about equality are postponed if the integration system is uncertain about equality. This uncertainty is stored using the probabilistic XML data model, making the integration process itself unattended. The amount of uncertainty arising from this integration can be large. We therefore introduce knowledge rules that help deciding on equality during the integration phase. Using these rules, integrated documents contain less uncertainty and are therefore smaller in size. We also introduced two measures with which the amount of uncertainty in the document can be quantified. Uncertainty density measures the amount of uncertainty in the database. The second measure, answer decisiveness, quantifies the ease with which most likely possibilities in query results can be chosen.

At a later stage, when the user is querying the information source, and therefore already actively using the system, feedback can be provided on query results. This feedback is explained in the same semantical setting as querying. Feedback statements can either be positive, i.e. the query result can be observed in the real world, or negative, i.e. the query result cannot be observed in the real world. We show that using this feedback technique, if used with caution, reduces the amount of uncertainty and lets the information source converge to a correctly integrated document. To measure the quality of query results, we adapted precision and recall for probabilistic data in a way that, for example incorrect answers with low probability do not have the same negative impact as incorrect answers with a high probability.

Samenvatting

Recentelijk is de vraag naar het ondersteunen van onzekere data toegenomen. Sensorapplicaties maken bijvoorbeeld gebruik van data die per definitie onzeker is, omdat de sensoren die de data leveren dit ook zijn. Huidige databasesystemen bieden geen ondersteuning voor onzekere data, anders dan dat deze onzekerheid als user-defined data opgeslagen kan worden. Op deze manier is de gebruiker zelf echter ook verantwoordelijk voor het correct verwerken van de onzekerheid.

In dit proefschrift presenteren we een nieuw, op XML gebaseerd datamodel, waarmee onzekerheid over elementen en subbomen kan worden opgeslagen. Het XML data model is op een dusdanige manier uitgebreid dat kansen aan elementen en subbomen kunnen worden gekoppeld. Ook kunnen afhankelijkheden en onafhankelijkheden van elementen en zelfs het al dan niet bestaan van deze elementen worden uitgedrukt. We geven een correcte semantiek voor het werken met de onzekerheden die gekoppeld zijn aan de data en laten zien hoe deze semantiek wordt gebruikt in het bevragen van de data.

Het probabilistische XML data model wordt toegepast in een informatie integratie applicatie. Keuzes over gelijkheid worden uitgesteld als het systeem onzeker is over deze gelijkheid. Deze onzekerheid wordt met behulp van kansen opgeslagen in het probabilistische XML datamodel, waardoor het integratie proces zelf autonoom is. De hoeveelheid onzekerheid die op deze manier ontstaat in het geintegreerde document, is groot. We introduceren daarom kennisregels, die kunnen helpen bij het nemen van de beslissing rondom gelijkheid. Door gebruik te maken van deze kennisregels, bevat het geintegreerde document minder onzekerheid en wordt het dus kleiner. We introduceren ook twee maten waarmee de hoeveelheid onzekerheid in het document kan worden gequantificeerd. Uncertainty density meet de hoeveelheid onzekerheid in het document, terwijl answer decisiveness het gemak waarmee het meest waarschijnlijke antwoord kan worden gekozen, aangeeft.

Op een later moment, wanneer de gebruiker van het systeem toch al gebruik maakt van het systeem, kan feedback op query resultaten worden

gegeven. Deze feedback wordt gebruik in dezelfde semantische setting als de bevragen van de data. Er zijn twee soorten feedback mogelijk, de eerste is positieve feedback waarbij aangegeven wordt dat het resultaat daadwerkelijk in de echte wereld bestaat. De tweede manier van feedback is negatieve feedback, waarmee aangegeven wordt dat het resultaat niet bestaat en dus ook niet in de database hoort te staan. We laten zien dat door het gebruik van deze feedback techniek, het geintegreerde document uiteindelijk zal convergeren naar een correct geintegreerd document. Om de kwaliteit van antwoorden te meten hebben we precision en recall aangepast om rekening te houden met de kansen die horen bij de data, zodat bijvoorbeeld foute antwoorden met een lage kans minder negatieve invloed hebben dan foute antwoorden met een hoge kans.

Index

SIKS Dissertatiereeks

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects
1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information
1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations
within the Language/Action Perspective
1998-4 Dennis Breuker (UM)
Memory versus Search in Games
1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting
1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling;
Automated modelling of Quality Change of Agricultural Products
1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets
1999-3 Don Beal (UM)
The Nature of Minimax Search
1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects
1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven
Specification of Network Information Systems
1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems
1999-7 David Spelt (UT)
Verification support for object database design
1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent
Mechanism for Discrete Reallocation.
2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance
2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management
2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie;
een procesbenadering en actorperspectief.
2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design
2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.
2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication
2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management
2000-8 Veerle Coup (EUR)
Sensitivity Analyis of Decision-Theoretic Networks
2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization
2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations,
Algorithms and Architecture
2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management
2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks
2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models
2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with
Instance-Based Boundary Sets
2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style
2001-6 Martijn van Welie (VU)
Task-based User Interface Design
2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization
2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.
2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models,
Views of Packages as Classes
2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice
BRAHMS: a multiagent modeling and simulation language
for work practice analysis and design
2001-11 Tom M. van Engers (VUA)
Knowledge Management:
The Role of Mental Models in Business Systems Design
2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis
2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections
2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval
2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining
2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments
inhabited by Privacy-concerned Agents
2002-06 Laurens Mommers (UL)
Applied legal epistemology;
Building a knowledge-based ontology of the legal domain
2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative
E-Commerce Ideas
2002-09 Willem-Jan van den Heuvel(KUB)
Integrating Modern Business Applications with Objectified Legacy Systems
2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble
2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications
2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems
2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications
2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and
Verifying Multi-Agent Systems
2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling
2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications
2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance
2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments
2003-02 Jan Broersen (VU)

Modal Action Logics for Reasoning About Reactive Systems
2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology
2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach
2003-06 Boris van Schooten (UT)
Development and specification of virtual environments
2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks
2003-08 Yongping Ran (UM)
Repair Based Scheduling
2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour
2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture
2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval
2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models
2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems
2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses
2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing
2003-18 Levente Kocsis (UM)
Learning Search Decisions
2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic
2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business
2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures
2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity
2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques
2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes
2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise
2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning
2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects
2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies
2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents
2004-13 Wojciech Jamroga (UT)

Using Multiple Models of Reality: On Agents who Know how to Play
2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium
2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining
2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning
2004-17 Mark Winands (UM)
Informed Search in Complex Games
2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models
2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval
2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams
2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications
2005-02 Erik van der Werf (UM))
AI techniques for the game of Go
2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language
2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data
2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing
2005-06 Pieter Spronck (UM)
Adaptive Game AI
2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems
2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications
2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages
2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry
2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes
2005-16 Joris Graaumans (UU)
Usability of XML Query Languages
2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components
2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks
2005-19 Michel van Dartel (UM)
Situated Representation
2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives
2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by
Exploiting Application Semantics
2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting
2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems
2006-04 Marta Sabou (VU)
Building Web Service Ontologies
2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines
2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools
for Graphical Service Modeling
2006-07 Marko Smiljanic (UT)
XML schema matching – balancing efficiency and effectiveness by means of clustering
2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web
2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor's Opinion
2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems
2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types
2006-12 Bert Bongers (VU)
Interactivation - Towards an e-cology of people, our technological environment, and the arts
2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents
2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain
2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks
2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device
2006-18 Valentin Zhizhkun (UVA)
Graph transformation for Natural Language Processing
2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach
2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining
2006-21 Bas van Gils (RUN)
Aptness on the Web
2006-22 Paul de Vrieze (RUN)
Fundaments of Adaptive Personalisation
2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web
2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources
2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC
2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval
2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories
2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval
2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures
2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach
2007-03 Peter Mika (VU)
Social Networks and the Semantic Web
2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach
2007-05 Bart Schermer (UL)