Addressing internal consistency with multidimensional conditional functional dependencies

Stefan Brüggemann

OFFIS - Institute for Information Technology
Escherweg 2
26121 Oldenburg
Germany
brueggemann@offis.de

Abstract

Conditional functional dependencies(CFDs) have recently been introduced as a novel approach for capturing the external consistency of relational data by comparing tuples. They define bindings of semantically related values that originate from flat domains. In real world applications often domains with multidimensional metadata have to be addressed and data are not only stored in relational databases. Since CFDs are not applicable to measure the internal consistency of a single tuple, we present an implementation of CFDs in ontologies that encapsulates CFDs from a physical data storage and allows for the detection of violations of the internal consistency of a tuple.

Our implementation of multidimensional CFDs (mdCFDs) encourages the creation of bindings between (root) nodes of tree-structures, which drastically reduces the number of bindings that have to be created. We enable the usage of mdCFDs for proactive error prevention and reactive detection of inconsistencies. The usage of multidimensional structures supports the suggestion of qualitatively good repair operations for invalid data. As CFDs are limited to the comparison of several tuples, our approach allows the detection of inconsistencies in a single tuple.

The method is designed for application scenarios in domains with existing domain knowledge that can be referenced as a closed discourse world. We show how this approach is being motivated by an epidemiological cancer registry.

1 Motivation

Data quality (DQ) is an important aspect in most application domains. Inconsistencies may occur in almost every data set. This is often the case in institutions of public health care systems and results in severe data quality problems [7, 1, 10, 11]. Especially in scenarios where data are being integrated from different data sources, the consistency of data has to be checked [13].

A well-known concept for the identification of inconsistencies is the validation of data against a defined set of consistency constraints like CFDs[3]. CFDs are intended for the identification of violations of the external consistency. They do not focus on the internal consistency of a single tuple. This is an essential aspect in health care environments, for instance. Each data set of a given patient has to be validated in order to select the proper operation or medication. In these environments data occur at different locations and are not limited to databases. Therefore a logically independent approach is needed for these domains. Not only data that are being integrated into a database have to be validated but also data that are being interchanged with partners. The presented approach in this paper is not intended to solve all DQ problems but moreover addresses the specific aspect of interal consistency.

Many domains like public health consist of knowledge with hierarchical structures like taxonomies. Sometimes this knowledge is already accessible in machine-readable forms like the Unified Medical Language System (UMLS)[1] or is being described as it is the case in multidimensional data models in data warehouse systems. Then this existing knowledge can be reflected as a starting point for the definition of consistency constraints.

Traditional functional dependencies (FDs) allow for the identification of integrity conflicts in databases. A relation R is free of conflicts if all tuple t of R meet a FD ϕ as follows: Consider a relation R:(A,B,C) and a FD $\phi = B \rightarrow C$. If two tuples $t_1 : (a, b, c)$ and $t_2 : (a, b, c_2)$ exist, then $c = c_2$.

International Conference on Management of Data
COMAD 2010, Nagpur, India, December 8–10, 2010

[1] www.nlm.nih.gov/research/umls/

CountryCode	ZIP	Street
44	-	-

Table 1: Pattern tableau Tϕ for CFD ϕ

In the last years the concept of FDs has been extended with the introduction of CFDs. This allows to define functional dependencies that only have to be fulfilled when specific conditions are met. These conditions are being declared in a pattern tableau t_p. Consider t_p containing a single tuple (42,'_') for the CFD ϕ. If two tuple $t_1 : (a, b, c)$ and $t_2 : (a, b, c_2)$ exist, then c must be equal to c_2 if and only if b=42.

CFDs can be applied for the reactive identification of integrity violations in multiple tuples in relational databases.

The paper at hand presents an approach that extends CFDs with multidimensionality and moreover enables the usage of CFDs for the proactive and reactive detection of inconsistencies in a single tuple. This means that mdCFDs can also be used for the prevention of errors and for the validation of a single tuple.

CFDs have been introduced as follows: CFD aim at capturing the consistency of data by incorporating bindings of semantically related values. A CFD ϕ on R is a pair $(R : X \rightarrow Y, Tp)$, where (1) X, Y are sets of attributes from attr(R), (2) $R : X \rightarrow Y$ is a standard FD, referred to as the FD embedded in ϕ; and (3) Tp is a tableau with all attributes in X and Y , referred to as the pattern tableau of ϕ where for each A in X or Y and each tuple $t \in Tp$, t[A] is either a constant 'a' in the domain dom(A) of A, or an unnamed variable '-'. The use of unnamed variables is an open world assumption: The example CFD [CountryCode, ZIP] $\rightarrow$ [Street] with the pattern tableau given in Table 1 defines that in the UK every ZIP exactly identifies a street (ZIP's in the UK have a quite different structure than in other countries as they uniquely identify a street). As a fundamental definition consistent data can be described as data that do not violate existing consistency constraints. On instance level, consistency is being defined as the legal combination of attribute values [2].

The original intention of CFDs was the checking of the external consistency of a given data set. Consider schema R with two tuples s_1=$(44, b_1, c_1)$ and s_2 $(44, b_2, c_2)$. Then s_1 and s_2 violate the given CFD ϕ with its pattern tableau if b_1=b_2 and c_1 != c_2. CFDs do not focus on the internal consistency of given data. This means that CFDs are not intended for validating whether the combination of (b_1, c_1) is valid for a single tuple s_1. This is caused by the usage of unnamed variables "-" in CFDs, which is an open world assumption and may be applicable for several domains and scenarios. Since the approach in this paper focuses on domains with a closed discourse world, we do not introduce these unnamed variables. This allows for the checking of a single tuple, not only the comparison of a tuple against another. Using ϕ and the pattern tableau given in T 1, with traditional CFDs it is not decidable whether a tuple t = (44, W2 2QB, Main Street) correctly describes main street in London, UK.

Related work [3] introduce CFDs and their applications. CFDs have some limitations:

- CFDs are only realized for flat domains. Hierarchical domains are not being addressed.
- CFDs are limited to relational databases. They are not logically independent from a physical data source. Since our approach has been implemented as a web service, it can handle data from any data source.
- CFDs are not portable. The defined rules cannot be exchanged with partners. The rules are not being described in a standardized format, so there is no possibility of exchanging them with partners. The approach presented in this paper utilizes ontologies and OWL for interchanging rules with partners.
- The concept of CFDs lacks of a user integration. It is not addressed how the user can be supported in defining CFDs. Our approach defines a domain specific language for this task.
- CFDs only focus on the external consistency of data due to the use of unnamed variables. This disables the measurement of the internal consistency of a single tuple.

The structure of the paper is as follows: In the next section we present an overview on related work. We define the requirements for a rule language that can be applied in health care scenarios and compare related work using these requirements. Subsequently we present typical metadata that has to be dealt with. Afterwards this paper introduces mdCFDs as a novel implementation of CFDs in ontologies enabling elegant access to complex domain specific knowledge, metadata annotation of CFDs, and the checking of the internal consistency of data with mdCFDs. We further introduce a domain specific language that is intended to support domain experts in modeling mdCFDs. The last contributions are an inference system for mdCFDs analog to CFDs and functions for the imputation of inconsistent tuple. Finally we evaluate the presented approach within a concrete scenario and finish the paper with a conclusion and an outlook on future work.

2 Related Work

We now present requirements for rule languages and use them to compare related approaches for the application for consistency checking in public health care systems.

2.1 Requirements for rule languages

The requirements for a rule language that can be applied in scenarios like health care environments have been introduced in [15] and are being described as follows:

- A rule definition language has to be **intuitive**. Domain experts should be able to learn it easily.
- A set of consistency constraints has to be **valid**ated not containing contradictory constraints but being self-consistent.
- The language should be **web-compatible**. It should be able to integrate classification systems from different vendors and namespaces. It should further enable the definition of consistency constraints for data that are stored among several data sources, for instance at different data producers like hospitals or medics, where one part of the data is located at one party and another part of the data is located at another party.
- Defined constraints should be **logically independent** from a physical implementation of instance data. The constraints can be interpreted as abstract knowledge that is not bound to a concrete database.
- Consistency constraints should be **portable**. Especially in health care environments data are often interchanged between data producers and data consumers. When the constraints against which the data have been validated are being exchanged as well the quality definitions can be provided to the recipient.
- A rule definition language should provide **advanced language constructs** that can be applied to hierarchical data. Experts should be enabled to classify constraints using the constructs "inheritance", "functionality", "transitivity", or "symmetry". These are the language constructs that ontology languages like the OWL (Web Ontology Language) provide.

In the following we give an overview on related work and compare them with these requirements.

Several constraint solutions exist in research and industry. Research approaches include edit/imputation systems [6] and CFDs [3], industrial solutions are business rule systems like ILog Rules[2] or Visual Rules[3].

Edit/Imputation-systems are often being applied in statistical domains. Edits are being formulated as error expressions like "age < 15, marital status = married". These edits are being applied to input data and identified inconsistencies are being corrected by the application of imputations. Imputations are intended to change the values of fields of data sets. This approach has three major goals:

- The data in each considered record should satisfy all edits by changing the fewest fields as possible (minimum change principle).
- Imputations should automatically be derived from edits. This assures that corrected records meet the defined edits.
- When a value in a field has to be changed, it should be replaced by a value that is chosen from the frequency distribution of the fields domain.

Winkler et al. [17] give an introduction into that area. Several implementations of this model exist [14] but further research is needed especially with the problem of error localization.

Edit/imputation systems can only be applied in scenarios where the correctness of data is not important and where data only have to be consistent. This is the case in the field of statistical analyses of survey data.

Business rule management systems (BRMS) are designed to describe business rules (BR). Consistency constraints and integrity rules like CFDs and edits are a special case of business rules [12], and therefore we discuss BRMS as well. Most of them have the same set of properties, and only vary in a few aspects. Most of them are intended to describe rules as if-then-else structures, switch-case structures, and support inheritance. Many are able to be integrated in development environments and can generate code. Most of them provide a visual language for the creation of rules.

We now compare the introduced approaches with the requirements for rule languages:

- **Intuitive**: Most of the described solutions are intuitive and easy to learn since the number of language constructs of the underlying grammar is manageable. Especially ILog-tools are intuitive as they provide a graphical interface.
- **Valid**: Contradictory definitions like " Neoplasms of prostate are only valid with gender male" and " Neoplasms of prostate are only valid with gender female " can be detected in all reflected solutions.
- **Web-compatible**: The approaches are not able to integrate existing classification systems or concepts that are distributed among certain parties or, for instance, namespaces.
- **Logically independent**: CFD are a novel approach for the definition of consistency constraints for data located in databases. ILog can be applied to several data sources, for instances to databases and XML-files. Most implementations of Edit/Imputation-systems require SQL.

[2] http://www.ilog.com/products/businessrules/
[3] http://www.visual-rules.de/

Requirement	CFD	E/I	BR
Intuitive	+	+	++
Valid	++	++	++
Web-compatible	– –	– –	–
Logically independent	– –	– –	+
Portable	– –	–	+
Inheritance	–	–	+
Functionality	+	+	+
Symmetry	–	–	–
Transitivity	+	–	–

Table 2: Comparison of related approaches with described requirements

- **Portable**: Since CFDs can only be defined in databases they are not logically independent and not portable. The latter means that constraints cannot be exchanged with data producers or consumers. Edit/Imputation-systems have to be rolled out in every scenario, but the set of edits can be reused.
- **Advanced language constructs**: All of the described solutions are able to support functionality, but none of them support symmetry. BR are the only solution that support inheritance and CFDs are the only concept with transitivity.

Table 2 displays this comparison and shows that none of the presented approaches meets all of the requirements and motivates the introduction of a novel approach based on CFDs.

3 Ontology based definition of multidimensional conditional functional dependencies

With the explanation of CFDs we depicted that there is a need for an ontology based representation. Existing classification schemata often have an underlying hierarchical or multidimensional structure. MADEIRA [16] is such an existing multidimensional data model that is being used in this work. The core concepts of MADEIRA are shown in figure 1 (green elements) and being described in the following.

Each dimension describes values of a specific domain which belong together. Examples are gender, products, or ICD. Each dimension consists of a hierarchy which defines a relation on aggregation layer. An aggregation layer groups semantically related values like all electronical products or all tumours that can occur in the mouth. Each aggregation layer consists of a set of categories. These categories are the concrete values of a reflected domain like specific products or specific tumours.

MADEIRA has been chosen because it enables the logical design of conceptional schemata and consists of a simple, manageable set of concepts with exactly one data structure with an explicitly and clearly defined semantic modeling of aggregation layer and hierachies of categorial attributes.

CFDs aim at describing restrictions between attributes like "City" and "ZIP". In our appproach these attributes are dimensions. Concrete attribute values like "New York" are either aggregation layer or nodes. This assumption has two requirements for the definition of multidimensional CFDs: First, dependencies between dimensions have to be created. Second, instances of CFDs that can be considered as entries of the pattern tableau have to be created between layer or nodes.

MADEIRA has been extended in our work with the definition of CFDs. Figure 1 shows the resulting metamodel.

The resulting metamodel consists of entities describing a multidimensional structure and defines entities for the creation of CFD constraints. As an example the ICD is a dimension containing a hierarchy with several aggregation layer like "chapter"or "group". Categories are identified as concrete codes like "C20.1". Each CFD representation can be defined with one or many dimensions on the left and on the right side and consists of a constraint table describing a pattern tableau containing constraints. Each side of the constraint can be built with instances from an aggregation layer or with concrete categories. A constraint type characterizes a constraint as being one of the following constraint types: inheritance, transitivity, functionality, and symmetry. An inheritance-constraint classifies all categories of an aggregation layer, not only the aggregation layer itself. Transitivity defines dependencies between constraints. A functional constraint defines that the left side of a constraint is only valid with the right side while a symmetric constraint defines a bijective constraint. This allows for the example definition of a CFD constraint "$ICD \rightarrow T, N, M$" where T, N, and M are further dimensions describing tumour, node, and metastasis. ICD is used on the left side and the others on the right. The constraint table contains constraints like "C02.1, T1, N2, M3" defining that a tumour can be classified using "C02.1" and the mentioned T-, N-, and M-codes. An inheritance and functionality constraint can define that, for instance, "C20" and all more specific categories are only valid with "T1, N2, M2". A validation of the CFD constraints themselves is needed to detect that an inconsistency has been defined with "M2" and "M3".

The ontology based definition of constraints allows the mapping of these constraints to any concrete database or file. This ontology then can be used to implement repair algorithms for consistency violations as described in [5].

Although measures are a basic concept of multidi-

[3] complex business processes can be described, but this is not neccessary for consistency checking

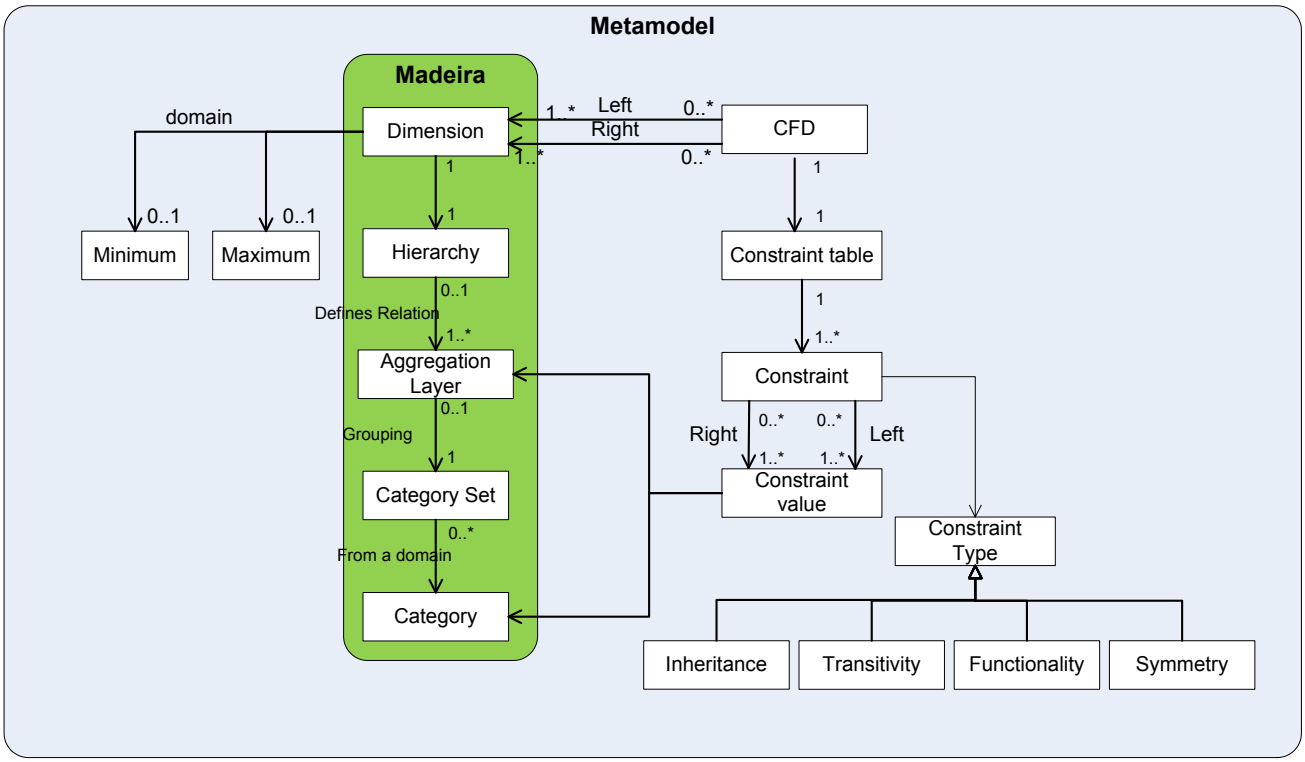

Figure 1: Metamodel of our definition of ontology based conditional functional dependencies

mensional data models we do not focus on them because we want to describe constraints and limitations of the multidimensional data structure. This structure can be considered as a closed discourse world in which the number of elements in a dimension is fixed. Measures do not meet these requirements as they originate from an open range of values. This would make it undecidable to classify a tuple as valid or invalid.

3.1 Description Logic

According to the formalism of CFDs (compare section 1) we are now able to present a formalism of mdCFDs. This formalism is not intended to be used by the domain expert. Regarding to the requirements for rule languages such a language should be "'intuitive"'. As domain experts typically are not familiar with description logics, we present a domain specific language in the next section.

mdCFDs are a multidimensional extension of CFDs and are based on the presented metamodel. A mdCFD ρ on an ontology O now is a pair $(O : X \rightarrow Y, T_p)$, where (1) X,Y are sets of dimensions from dim(O). dim(O) consists of all OWL-concepts that are $< rdfs : subClassOf >$ of a $< cfd : dimension >$. (2) $O : X \rightarrow Y$ is a standard FD, referred to as the FD embedded in ρ; and (3) T_p is a tableau with all attributes in X and Y, referred to as the pattern tableau of ρ where for each A in X or Y and each tuple $t \in T_p$, t[A] is a constant 'a'. The constant 'a' is either a $< cfd : layer >$ or a $< cfd : category >$ of a $< cfd : dimension >$ A.

3.2 Domain specific modeling

The introduced formalism in description logic from the previous section is not intended to be used by domain experts. An analyst or an expert does not want to use a language like that. Therefore we followed the approach of software language engineering. We developed a user-centered language in order to raise the level of abstraction for domain experts and created a graphical domain specific language (DSL) [8]. The main benefit of a DSL is that such a language is designed for a concrete use case and contains only relevant aspects of a domain. In our case the domain is the definition of mdCFDs for the DQM-task "'consistency control"'. According to [9], six parts are needed to define a DSL:

- An abstract syntax to define the elements of the language.
- One or more concrete syntax models to define each of the elements.
- A model transformation for the translation of the concrete elements into abstrach ones.
- Optionally a description of the meaning of the system.
- Optionally a definition of the languages that are needed to use the DSL.

- Optionally an interface definition, to make the DSL reusable.

Domain experts are needed to create and maintain domain specific constraints like mdCFDs. In most cases they are not familiar with ontology based modeling tools and do not want to create ontologies, but rules. The main idea is that transformations are being defined that convert from the user-DSL into the description logics formalism. The concrete syntax of the DSL is very similar to the metamodel we defined in figure 1 and consists of the same entities (compare [4] for a detailed explanation). An example of this DSL based on that concrete syntax has already been introduced in [13].

3.3 Consistency checking

In contrast to traditional CFDs, where algorithms for consistency checking had to be developed, our ontology based approach does not require any consistency checking algorithms. We rather are able to use an OWL-reasoner like pellet[4] for that task. A reasoner can decide whether an ontology is consistent or not. As our ontology is being implememented with language constructs of OWL-DL (description logic), it is decidable. Input data that has to be checked is being considered as an instance of the ontology. Then a reasoner-based consistency check can classify a tuple as consistent or not.

3.4 An inference system for mdCFDs

An inference system for CFDs has already been defined with the introduction of CFDs. We now show that this inference system can also be used by mdCFDs with modifications. The major differences of mdCFDs against CFDs are the usage of multidimensional structures with layer and nodes, the limitation of using only bound variables (without "-"), and the constraint types symmetry, transitivity, inheritance, and functionality. This has some implications on the inference system:

- Unbound variables act as placeholder for concrete values in CFDs. We have already shown that this concept cannot be used when focusing on the internal consistency of a single attribute. This is not a limitation to the application of inference rules to mdCFDs as simply the concrete values for an unbound variable have to be used in the pattern tableau.
- Entries in the pattern tableau t_p of a mdCFD $\phi = X \rightarrow Y$ may be layer or node and denoted as $t[X](l)$ or $t[X](n)$. It is obvious that nodes behave like normal variables in respect to CFDs and that layer can be interpreted as sets of nodes or as iterations of their children, for the purpose of inference.
- Only layer that are labeled with "inheritance" need to be replaced with iterations of their children.
- The inference rule "transitivity" is only applicable for mdCFDs that are labeled as "transitivity".
- Symmetric mdCFDs are CFDs of the form $\phi = (X \leftrightarrow Y, t_p)$. These are being interpreted as $\phi_1 = (X \rightarrow Y, t_{p_1})$ and $\phi_2 = (Y \rightarrow X, t_{p_1})$ and result in a new inference rule IR8: If $\phi = (X \leftrightarrow Y, t_p)$, then $\phi_1 = (X \rightarrow Y, t_{p_1})$ and $\phi_2 = (Y \rightarrow X, t_{p_1})$.
- Functionality describes that exactly one entry is allowed on the right hand side of a given entry from a left hand side. As the inference rules for CFDs are being built with pattern tableaus that only consist of a single pattern tuple t_p without the loss of generality, it is not a limitation to the appliance of the inference rules to mdCFDs.

These implications lead to the resulting inference rules shown in figure 2. IR1 to IR5 are similar to the inference rules defined for CFDs. FD6 of the inference system of the CFDs is not required for mdCFDs due to the removal of unbound variables. IR6 and IR7 are equivalent to the FD7 and FD8 of the inference system for CFDs. IR8 is a newly introduced inference rule that expresses the symmetry property that can be defined for mdCFDs.

3.5 Inconsistency correction

Existing approaches like CFDs correct identified inconsistencies by automatically editing attribute values. The aim of these approaches is fulfilling the set of defined consistency constraints. Our approach differs from these as we do not automatically apply corrections to tuple but rather provide correction suggestions to the user. The user might be the user of a data production system or the initiator of a data integration process. We assume that the user is the only one that can decide about the correctness of a tuple in respect to a real world entity. Existing approaches would ensure the consistency of a tuple but would ignore the correctness. This results in an approach where we can suggest possible valid corrections of a tuple and let the user select one of these corrections.

The ontology based definition of mdCFDs allows for the suggestion of semantically related corrections. Assume for instance that a tuple (C20.2, T_1, N_1,M_1) has been identified as inconsistent because there is no entry in the pattern tableau of ϕ (compare section 1). A semantically related correction suggestion makes use of the hierarchical structure of the dimensions a mdCFD consists of. This results in searching for an entry in the pattern tableau that contains either a sibling of

[4] http://clarkparsia.com/pellet

IR1: If $A \in X$, then $(X \rightarrow A, t_p)$, where $t_p[B] = 'b'$ for all $B \in X \cup \{A\}$.

IR2: If $(R : X \rightarrow A, t_p)$ and $B \in attr(R)$, then $(R : [X, B] \rightarrow A, t_p)$, where $t'_p[B] = 'b'$ and $t'_p[C] = t_p[C]$ for each $C \in X \cup A$.

IR3: If $(1)(X \rightarrow A_i, t_i)$ such that $t_i[X] = t_j[X]$ for all $i, j \in [1, k]$, (2) $([A_1, ..., A_k] \rightarrow B, t_p)$ and moreover, (3) $(t_1[A1], ..., t_k[A_k]) \preceq t_p[A_1, ..., A_k]$, then $(X \rightarrow B, t'_p)$, where $t'_p[X] = t_1[X]$ and $t'p[B] = t_p[B]$.

IR4: If $([B, X] \rightarrow A, t_p), t_p[B] = 'b'$, and $t_p[A]$ is a constant, then $(X \rightarrow A, t'_p)$, where $t'_p[X \cup A] = t_p[X \cup A]$.

IR5: If $([B, X] \rightarrow A, t_p)$ and $t_p[B] = 'b'$, then $([B, X] \rightarrow A, t'_p)$, where $t'_p[C] = t_p[C]$ for each $C \in X \cup A - B$, and $t'_p[B] =' b'$ for some $'b' \in dom(B)$.

IR6: If (1) $\Sigma \vdash_I ([X, B] \rightarrow A, t_i)$ for $i \in [1, k]$, (2) $\mathrm{dom(B)} = b_1, ..., b_k, b_{k+1}, b_m$, and $(\Sigma, B = b_l)$ is not consistent except for $l \in [1, k]$, and (3) for $i, j \in [1, k], t_i[X] = t_j[X]$, and $t_i[B] = b_i$, then $\Sigma \vdash_I ([X, B] \rightarrow A, t_p)$ where $t_p[B] =' b'$ and $t_p[X] = t_1[X]$.

IR7: If $B \in attr(R)$, $\mathrm{dom(B)} = b_i | i \in [1, m]$, and $(\Sigma, B = b_l)$ is consistent only for b_1, then $\Sigma \vdash_I (R : B \rightarrow B, (b, b_1))$.

IR8: If $\phi = (X \leftrightarrow Y, t_p)$, then $\phi_1 = (X \rightarrow Y, t_{p_1})$ and $\phi_2 = (Y \rightarrow X, t_{p_1})$

Figure 2: Inference rules for mdCFDs

the node "C20.2" (like "C20.1" or "C20.9") or a parent or child node of "C20.2" (like "C20" or "C20.21"). The same approach holds for the imputation of values from the right hand side of ϕ like "T", "N", and "M".

This motivates the definition of the following functions for the generation of correction suggestions:

- next-sibling: This function replaces an attribute value with an existing sibling of a node.
- first-child: An attribute value is being replaced with an existing child node, if and only if the reflected attribute value is an aggregation regarding to the definition in the metamodel as shown in figure 1.
- parent: The parent node of the given node is being used for the replacement of a given inconsistency, if and only if the node has a parent.

3.6 Summary

We now summarize the main advantages of mdCFDs and show why they are more powerful than traditional CFDs:

- CFDs focus on the external consistency of tuples by comparing them. mdCFDs are able to reflect the internal consistency of a single tuple. CFDs are not able to answer the question whether a given tuple is correct. They can only answer whether CFDs between more than one tuple are fulfilled or not.
- CFDs do not focus on the question how a pattern tableau is being defined. They do not provide a user integration or other techniques. Since such a tableau might be quite large in real world scenarios, the approach introduced in this paper provides solutions for that problem using a DSL. This DSL can be used by domain experts and allows for an intuitive definition of such a tableau.
- The language constructs provided through that DSL allow for the definition of a large number of tableau entries with little effort, for instance using "inheritance", which can define entries for an entire subtree with one click.
- Domain specific knowledge is often being represented by multidimensional metadata. It is self-evident to use that knowledge in that structure. Furthermore, in data warehouse environments, dimensions and multidimensional models often already exist and can be reused for DQM.

4 Evaluation

The presented approach has been introduced in the cancer registry of lower saxony[5]. This registry collects cancer cases and analyzes them for special purposes. Data from several data providers are being used, for instance from pathologists, medics, or hospitals. These data contain a couple of medical information like the age and sex of a patient, the kind of cancer, the localization of a tumour, and other. Several consistency constraints can be defined in that registry like "$\phi = ICD \rightarrow T, N, M$". Using the introduced inference system for mdCFDs, ϕ is equal to the set of constraints $\Sigma =$ ("$\phi_1 = ICD \rightarrow T$","$\phi_2 = ICD, T \rightarrow N$,"$\phi_3 = ICD, T, N \rightarrow M$"). A large number of instances of this consistency constraint exist. The consistency constraints and the instances are being persisted in a triple store, which is a kind of storage for ontologies. This storage serves the domain knowledge that is needed for the detection and removal of inconsistencies.

[5] http://www.krebsregister-niedersachsen.de

5 Conclusion and future work

In this paper we have introduced a novel approach that enhances the existing concept of conditional functional dependencies with mulitidimensionality. Several application domains consist of hierarchical metadata and we have presented mdCFDs that enable the usage of these hierachies.

We have further implemented this approach in an ontology-based way using OWL-DL. Since CFDs are limited to relational databases, this limitation is not longer the case for mdCFDs since our ontology-based realization is logically independent from a physical data source. Using OWL-DL also enabled the definition of a very convenient way of checking the consistency of input data, since this task can automatically be performed using a reasoner. This means that we did not have to implement specific consistency checking algorithms.

The usage of ontologies also allows the exchange of defined mdCFDs, for instance with partners. These mdCFDs can be interpreted as a way of knowledge, like domain knowledge or asset that a company possesses. The ability of interchanging this knowledge is an important aspect in real world scenarios.

We have further introduced a domain specific language that provides an easy way of defining mdCFDs and especially supports the definition of those between multidimensional structures.

As traditional CFDs have originally been used for the detection and removal of inconsistencies only between several tuples, they were not able to detect inconsistencies in a single tuple. This is a serious limitation in most real world situations and therefore we designed the concept of mdCFDs in a way that avoids this limitation. This leads us to the possibility of using mdCFDs in a *proactive* and *reactive* way. The proactive way means that consistency checks can be performed for a single tuple during the data production process.

Future work will focus on a process model that provides a structured approach for the integration of a consistency checking infrastructure in DQM environments.

References

[1] L. Bailey. Health care corner: The costs of poor quality health care. *Publications of the Institute of Applied Research*, 49, 2003.

[2] C. Batini and M. Scannapieco. *Data Quality.* Springer-Verlag, Berlin Heidelberg, 2006.

[3] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. *International Conference on Data Engineering*, 0:746–755, 2007.

[4] S. Brüggemann and F. Grüning. *Networked Knowledge - Networked Media: Integrating Knowledge Management, New Media Technologies and Semantic Systems*, chapter Using Ontologies Providing Domain Knowledge for Data Quality Management. Springer, 2009.

[5] W. Fan, F. Geerts, and X. Jia. Semandaq: a data quality system based on conditional functional dependencies. pages 1460–1463. VLDB Endowment, 2008.

[6] I. Fellegi and D. Holt. A systematic approach to automatic edit and imputation. *Journal of the American Statistcal Assocation*, 71:17–35, 1976.

[7] T. Johnsen. Weaknesses in code quality in hospitals. Technical report, Riksrevisjonen, 2006.

[8] S. Kelly and J.-P. Tolvanen. *Domain-Specific Modeling: Enabling Full Code Generation.* John Wiley & Sons, 2008.

[9] A. Kleppe. *Software Language Engineering.* Addison-Wesley, 2009.

[10] L. T. Kohn, J. M. Corrigan, and M. S. Donaldson, editors. *To Err is Human. Building a Safer Health System.* Committee on Quality of Health Care in America, 2000.

[11] D. MacDonald. Data quality management: Oft-overlooked key to affordable, high quality patient care. Technical report, HCT Project, 2004.

[12] R. G. Ross. *Principles of the Business Rule Approach.* Addison-Wesley, 2003.

[13] Y. Teiken, S. Brüggemann, and H.-J. Appelrath. Interchangeable consistency constraints for public health care systems. In *Proceedings of the 25th Annual ACM Symposium on Applied Computing*, volume Volume 2 of 3, pages 1411–1416. ACM, 3 2010.

[14] T. D. Waal and R. Quere. A fast and simple algorithm for automatic editing of mixed data. *J. Official Statist*, 19, 2003.

[15] P. Watson. Formal languages for expressing data consistency rules and implications for reporting of quality metadata. In I. W. Group, editor, *5th International Symposium on Spatial Data Quality, ISSDQ 2007, Enschede, NL*, volume II/7, 2007.

[16] F. Wietek. Modelling multidimensional data in a dataflow-based visual data analysis environment. In *Advanced Information Systems Engineering*, volume 1626/2010, pages 149–163. Springer Berlin / Heidelberg, 1999.

[17] W. E. Winkler. Methods for evaluating and creating data quality. *Information Systems*, 29, 2004.